



# SE LEVEL II

RTE IV

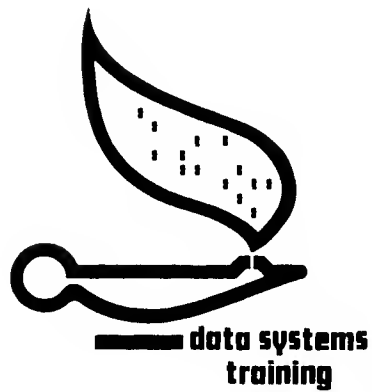


student  
workbook



# SE LEVEL II

## RTE IV







RECOMMENDED COURSE OUTLINE FOR SE Level II COURSE

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:00	Introduction	Review Labs	Resource Numbers LU Locks	Review Hw #3	Review Hw #2,4
9:00	Hardware Overview	Operator Requests -Trace "ON,XYZ" From Keyboard to \$XEQ		Re-Entrant Processing -LIBR/LIBX -REIO SAM -Users of Sam	Power Fail System Library Utilities
			COFFEE BREAK		
10:00	RTE Overview	Program Dispatching Partition Assignment	Review Hw #1 Program States -State Diagram -\$List	-SAM Management	Performance Measurement
11:00	RTE Modules			I/O Drivers -Initialization -Continuation -Completion -Privileged	Lab Seminar
12:00					
			LUNCH		
1:00					
2:00	DMS -Phy./Log. Memory -RTE Maps Boot Process -Trace From Front Panel Thru \$STRT	I/O Processing Over- view  Exec Calls -Trace Exec 2 Call from MP Thru I/O	TBG Time Tick -Trace From Interr. to \$XEQ		EMA -EMA in Fortran -EMA in Assembler -EMAST, MMAP -.EMAP,.EMIO
			COFFEE BREAK		
3:00	CMM4/DBUGR	Completion  Parity Errors	Class I/O MTM -Trace From Keyboard Thru R\$PN\$	Lab	Exam
4:00	Lab	Lab	Lab		
5:00					



## PREFACE

This student work book is to be used with the SE Level II training course and consists of the following sections:

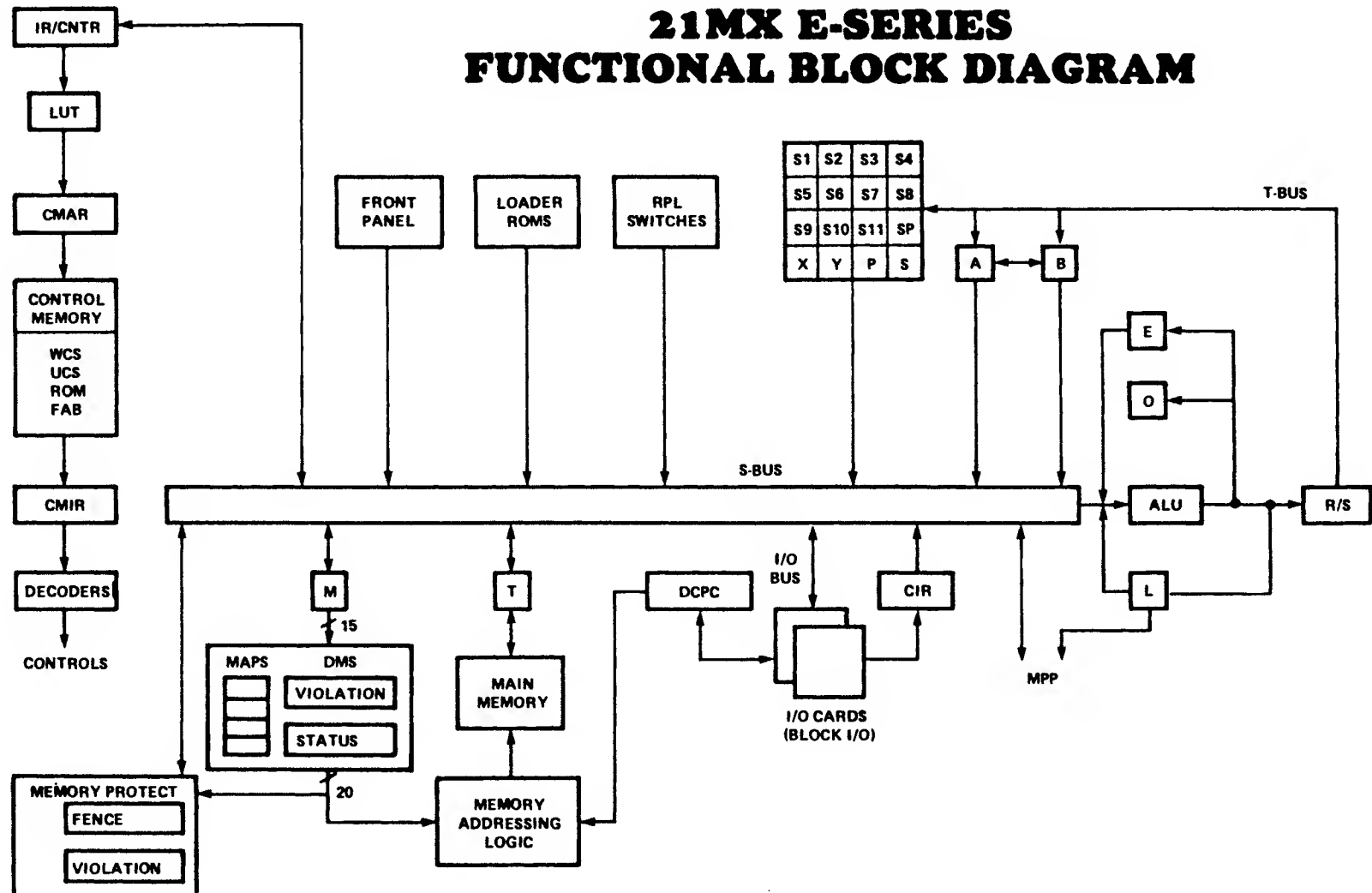
SECTION NUMBERS	TOPICS
-----	-----
1	HARDWARE OVERVIEW
2	RTE OVERVIEW
3	RTE MODULES
4	DMS
5	BOOT PROCESS
6	DBUGR/CMM4
7	OPERATOR REQUESTS
8	PROGRAM DISPATCHING/PARTITION ASSIGNMENT
9	I/O PROCESSING
10	EXEC CALL PROCESSING
11	PARITY ERRORS
12	RESOURCE NUMBERS
13	LOGICAL UNIT LOCK
14	PROGRAM STATES
15	TBG TIME TICK
16	CLASS I/O
17	MULTI-TERMINAL MONITOR (MTM)
18	RE-ENTRANT PROCESSING
19	SYSTEM AVAILABLE MEMORY (SAM)
20	I/O DRIVERS
21	POWER FAIL
22	SYSTEM LIBRARY
23	UTILITIES
24	PERFORMANCE MEASUREMENT
25	EMA
A	SYSTEM TABLES/LISTS
H	HOMEWORK & LAB ASSIGNMENTS



## **HARDWARE OVERVIEW**



# 21MX E-SERIES FUNCTIONAL BLOCK DIAGRAM



## 21MX-E SECTIONS

- CONTROL PROCESSOR  
Controls all other sections with microinstructions
- ARITHMETIC/LOGIC  
ALU, L, R/S, O, E, A, B, and 16 RAM registers
- MAIN MEMORY  
M and T registers
- INPUT/OUTPUT  
CIR and I/O cards
- OPERATOR PANEL  
Microprogrammed front panel
- MEMORY PROTECT
- DYNAMIC MAPPING  
Optional
- DUAL CHANNEL PORT CONTROLLER  
Two channels assignable



## INTERRUPT SYSTEM

Vectored priority interrupt system with distinct interrupt levels. Each level is associated with a corresponding interrupt location (trap cell) in memory.

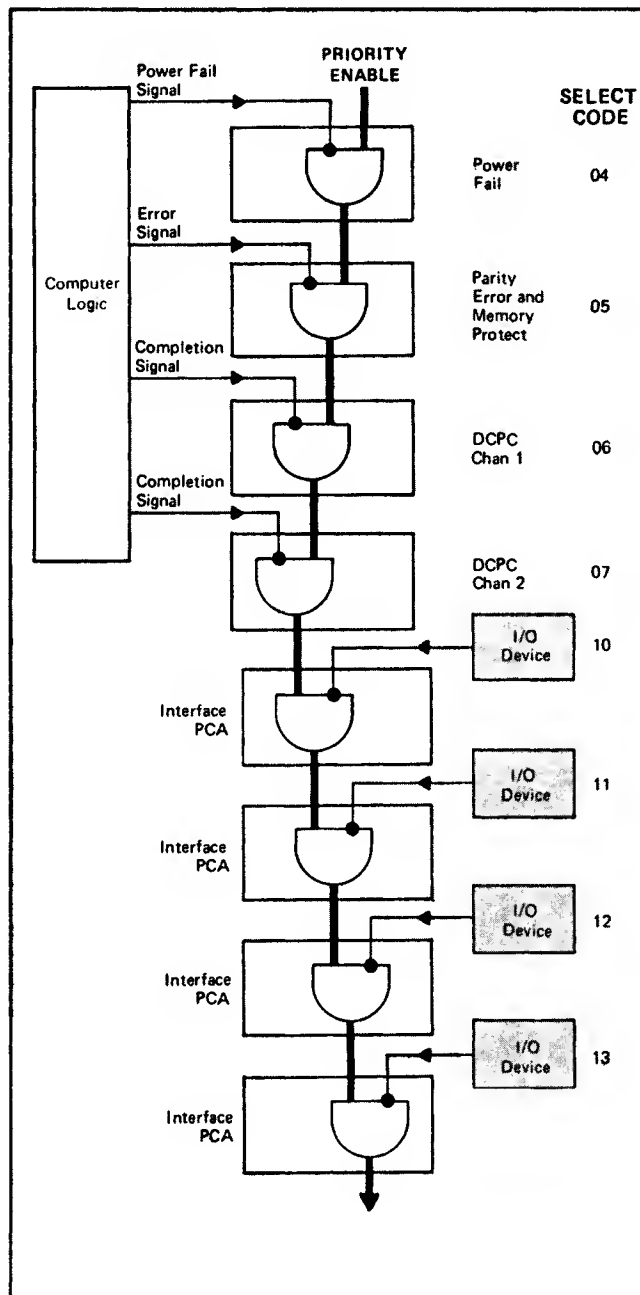
CHANNEL OCTAL	INTERRUPT LOCATION	ASSIGNMENT
00	NONE	INTERRUPT SYSTEM DISABLE/ENABLE
01	NONE	SWITCH REGISTER OR OVERFLOW
02	NONE	DCPC CHANNEL 1 INITIALIZE
03	NONE	DCPC CHANNEL 2 INITIALIZE
04	04	POWER FAIL INTERRUPT/CIR
05	05	MEMORY PARITY/MEMORY PROTECT/DMS INTERRUPT
06	06	DCPC CHANNEL 1 COMPLETION INTERRUPT
07	07	DCPC CHANNLE 2 COMPLETION INTERRUPT
10	10	I/O DEVICE (HIGHEST PRIORITY)
thru	thru	thru
77	77	I/O DEVICE (LOWEST PRIORITY)

An interrupt causes the instruction in the "TRAP CELL" to be executed. The interrupt select code is stored the CIR.

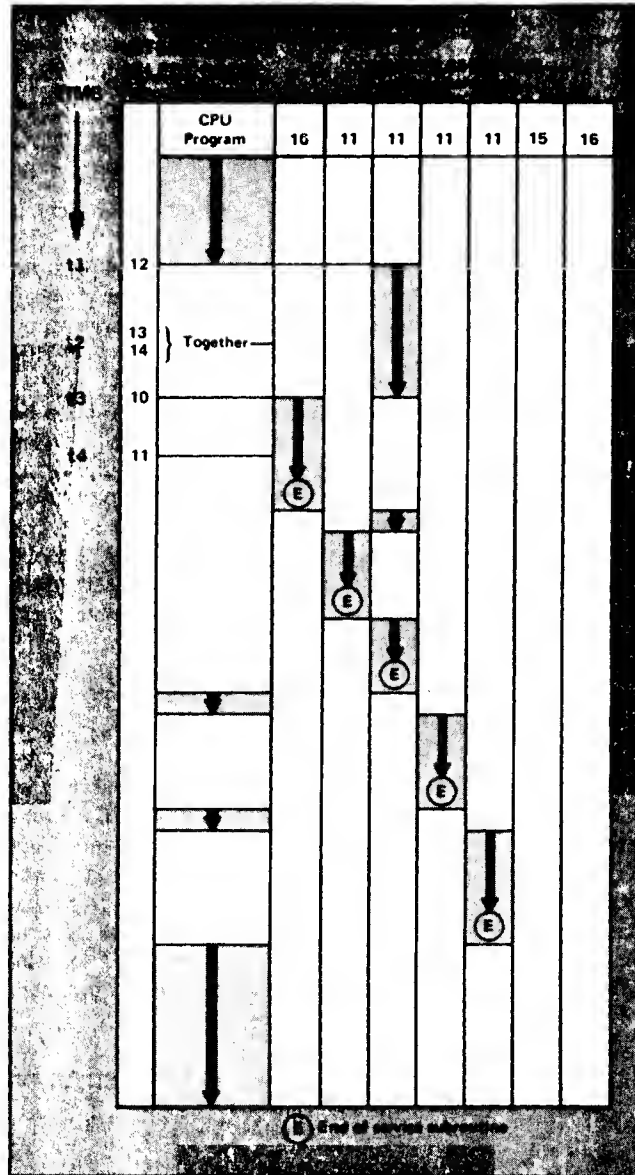
## INTERRUPT PRIORITY

Interrupt priority decreases with increasing select code.

A series-linked priority structure allows any device to interrupt and hold off interrupts from devices with higher select codes (SC).



# EXAMPLE INTERRUPT SEQUENCE



NOTE: RTE turns off the interrupt system while a service routine or driver is executing unless a privileged interface card is present.

## REQUIREMENTS FOR AN INTERRUPT

1. Interrupt system is enabled. (STF 0)
  2. Device flag flip-flop is set.
  3. Device control flip-flop is set. (STC)
  4. Device has priority.
  5. Interrupt recognition is enabled.
- 
- Interrupt system is enabled/disabled with a STF 0/CLF 0.
  - Flag bit is used by a device to request service from the computer.
  - Control bit is used to enable/disable a device.
  - Interrupts are inhibited until the succeeding instruction is executed for:

JMP indirect	STC	CLC	SFS (E series)
JSB indirect	STF	CLF	SFC (E series)

## DCPC

Provides a direct path, software switchable, between memory and I/O devices. Two DCPC channels are available that operate on a cycle-stealing basis in the following priority:

DCPC1  
DCPC2  
CPU

### DCPC OPERATION:

1. Initialize the DCPC channel with the I/O devices select code, transfer direction, buffer address, and word count.
2. Data transfer is accomplished on a word-by-word basis under automatic control of the DCPC hardware. This eliminates interrupting to a device driver after each word transfer.
3. DCPC completion interrupt is generated when the data transfer is finished. Optionally, the device also generates an interrupt upon completion of the data transfer.

## MEMORY PROTECT

Protects a selected block of memory from a settable fence address downward. An interrupt on select code (SC) 5 is generated when:

- One of the following instructions directly or indirectly modifies or enters a memory location below the fence:

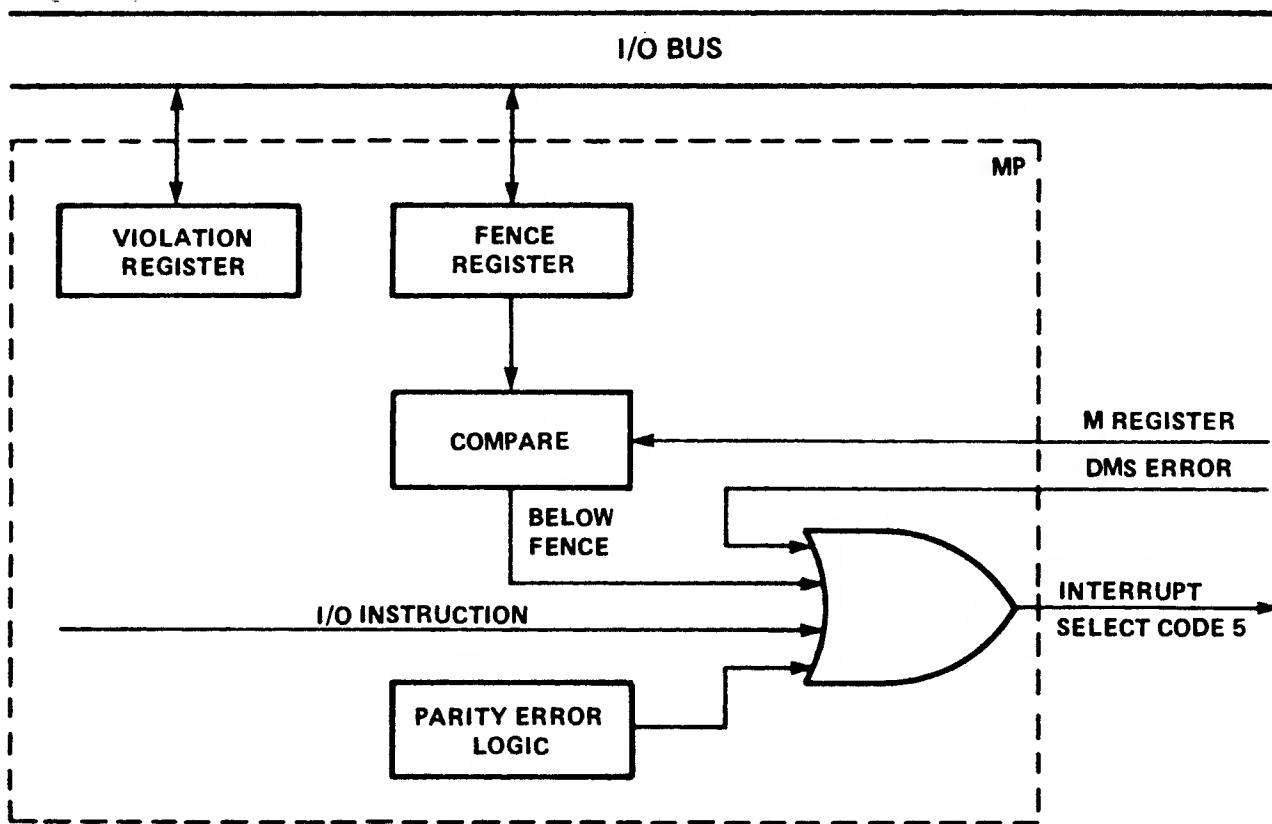
DST, ISZ, JLY, JMP, JPY, JSB, MVB, MVW, SAX, SAY, SBX, SBY, STA, STB, STX, STY

- Any I/O instruction is attempted except instructions to select code (SC) 1.
- HLT instruction is attempted.
- A DMS instruction is attempted. (Many of the DMS instructions are allowed if the system map is enabled.)

The violation register will contain the address of the illegal instruction.



## MEMORY PROTECT LOGIC



THE CAUSE OF A SC (SELECT CODE) 5 INTERRUPT CAN BE DETERMINED BY:

- MP    – BIT 15 OF VIOLATION REGISTER IS CLEAR\*
- PE    – BIT 15 OF VIOLATION REGISTER IS SET\*
- DMS   – FLAG ON SC 5 IS SET (SFS 5 OR SFC 5)

\*VIOLATION REGISTER IS READ WITH: LIA/B 5





**RTE  
OVERVIEW**



		RTE IV		
PROG TYPES	*	-----*		
0	*	DRIVER PARTITION #n	*	
	*	-----*		
	*	.	*	
	*	.	*	
	*	.	*	
	*	-----*		
0	*	DRIVER PARTITION #2	*	
	*	-----*		
(16)	*	SAM (\$CNFG)	*	
	*	-----*		
	*	PERR4	*	
	*	-----*		
	*	OCMD4	*	
	*	-----*		
	*	\$ALC	*	
	*	-----*		
	*	SCHD4	*	
	*	-----*		
0	*	\$TRN4	*	MEMORY
	*	-----*		RESIDENT
	*	EXEC4	*	SYSTEM
	*	-----*		
	*	RTIO4	*	
	*	-----*		
	*	\$ASC4	*	
	*	-----*		
	*	RTIME	*	
	*	-----*		
	*	DISP4	*	
	*	-----*		
	*		*	TAT,MATA,MPFT,
	*		*	KEYWORD,ID
13	*	TABLE AREA II	*	SEG, CLASS,
	*		*	RN'S, LU SWITCH
	*	-----*		
0	*	SYSTEM DRIVER AREA	*	
	*	-----*		
	*	BG COMMON	*	
	*	-----*		
	*	FT COMMON	*	
	*	-----*		
30	*	SSGA	*	
	*	-----*		
0	*	DRIVER PARTITION #1	*	
	*	-----*		
	*	sam	*	INTERRUPT,
	*		*	DRT,DVMP,
15	*	TABLE AREA I	*	EQT,TB3X
	*	-----*		2000
	*	SYSTEM BASE PAGE	*	
	*	-----*		0

## RTE IV

```

PROG TYPES *-----* 1000K MAX
*
*
*
2 3 4 5 * DISC RESIDENT PARTITION *
10 11 12 18 * #n *
19 20 26 27 *-----*
28 * DISC RES. PART. n BASE PAGE *
*-----*
*
-----
-----
*
*-----*
*
* DISC RESIDENT PARTITION *
* #1 *
*
*-----*
*
* DRP 1 BASE PAGE *
*
*-----*
*
*
* SYSTEM AVAILABLE MEMORY *
* EXTENSION *
* (SAM) *
*
*-----*
*
*
*
*
* MEMORY *
* RESIDENT *
*
*
*
*-----*
*
* MEMORY RESIDENT LIBRARY *
*
*-----*
*
* MEMORY RESIDENT BASE PAGE *
*-----*
*
*

```

## DRIVER PARTITIONS

Driver partitions contain one or more device drivers. All driver partitions are the same size and only the partition containing the driver currently being used is included in the user's logical map. The minimum partition size is two pages.

### SYSTEM DRIVER AREA (SDA)

This area contains all drivers not allocated to a driver partition. SDA is not included in the large BG disc resident map and is optional in the memory resident map. Drivers should be put into SDA for the following reasons:

- A. Drivers greater than 2K words would be included in SDA to reduce the size of driver partitions and thus increase the potential size of type 4 programs.
- B. Privileged drivers are put into SDA since they must always be present in the system map. This results from the fact that an interrupt from a privileged device enables the system map and then jumps directly to the privileged section of the driver.
- C. Drivers that do their own mapping must also be put into SDA. Since RTE enters self-mapping drivers with the system map enabled these drivers (like privileged drivers) must always be present in the system map.

There is a restriction to placing drivers in SDA that do not do their own mapping. Drivers in SDA may only be used for class I/O or buffered output requests from programs that do not have SDA in their logical maps. This includes all type 4 programs and possibly memory resident programs.

SYSTEM DISC		
LU2		
*-----*		255
* FMP DIRECTORY *		
*-----*		
*-----*		
*-----*		
-----		
-----		
*-----*		
* FMP-OWNED TRACKS *		
* (FMGR) *		
*-----*		
*-----*		
*-----*		
*-----*		
*-----*		
*-----*		
*-----*		
*-----*		
*-----*		100
*-----*		
* SYSTEM AVAILABLE TRACKS* *		
*-----*		
* SWAPPING *		
* USER TRACKS *		
* GLOBAL TRACKS *		
* ON-LINE LOADED PROGS *		
*-----*		25
*-----*		
* ENTRY POINT DIRECTORY *		
*-----*		
*-----*		
* DISC RESIDENT LIBRARY SUBROUTINES *		
* (TYPE (6), 7, AND 14) *		
*-----*		
*-----*		
* GENERATION LOADED PROGRAMS *		
*-----*		
*-----*		
* PTE SYSTEM *		
*-----*		
* SYSTEM BASE PAGE *		
*-----*		
* BOOT EXTENSION *		
*-----*		0

\* SYSTEM ALLOCATES FROM THE TOP TRACK DOWN; USER ALLOCATES FROM THE BOTTOM TRACK UP

## ENTRY POINT DIRECTORY

- Addresses of memory resident system modules, tables, lists, and drivers

examples:	\$XEQ	\$CIC	EXEC
	\$MATA	\$IOUP	I.05
	\$XSIO	\$RQST	C.05
	\$LIST	\$ALC	I.12

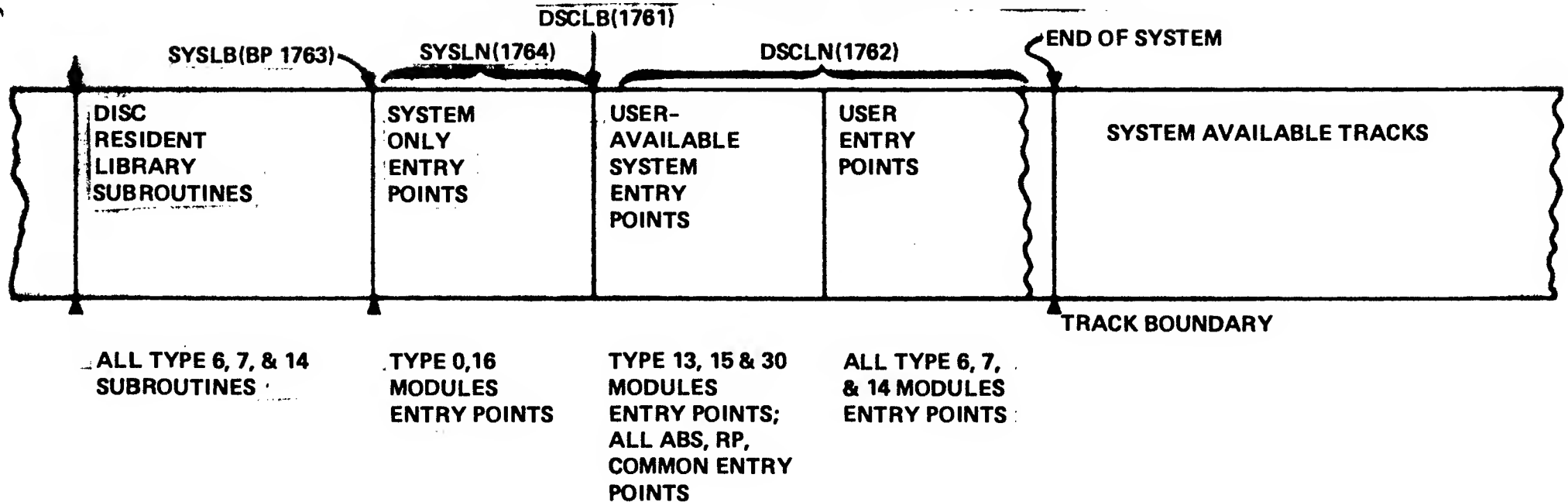
- Track and sector of disc resident library subroutines

examples:	RNRQ	READF	COS
	RMPAR	ABREG	ABS
	PARSE	DBUGR	DMAX1
	CLOSE	REIO	CSQRT

- Microcode replacement values

examples:	FLOAT	.MVW	.DLD
	IFIX	.DIV	.GOTO

# ENTRY POINT DIRECTORY



**NOTE:** SYSLN & DSCLN CONTAIN THE NUMBER OF 4-WORD ENTRIES

## LIBRARY ENTRY POINTS LIST

**FORMAT:**

word 1	name 1,2
word 2	name 3,4
word 3	name 5, flag bits
word 4	value

**flag bits:**

000	memory resident entry point
001	disc resident subroutine
010	common entry point
011	absolute
100	replace



# LGTAT PRINTOUT

## (LONG FORM)

:RU,LGTAT,,1

### TRACK ASSIGNMENT TABLE

& =PROG ^ =SWAP

TRACK	0	1	2	3	4	5	6	7	8	9
0	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	JOB &	AUTOR&
10	LOADR&	GASP &	FMGR &	FMGR0&	FMGR2&	FMGR4&	FMGR6&	FMGR8&	TVST4&	ASMB0&
20	XREF &	SYSTEM	F4.4 &	F4.0 &	DRSTR&	DRSTR&	DSAVE&	LIBRY	LIBRY	
30	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY
40	LIBRY	ENTS	SYSTEM	MEAS7&	FMGR	--	GLOBA	--	--	--
50	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--
80	--	--	--	--	--	--	--	--	--	--
90	--	--	--	--	--	--	--	--	--	--
100	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
110	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
120	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
130	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
140	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
150	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
160	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
170	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
180	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
190	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
200	FMP	FMP	D.RTR							

SYSTEM OWNED  
TRACK

GLOBALLY OWNED  
TRACK

SYSTEM  
ENTRY POINTS

RELOCATABLE  
SYSTEM LIBRARY

LOADED PROGRAMS  
"F4.4" AND "MEAS7"  
STORED HERE (PROGRAM  
NAME FOLLOWED BY "&")

# AUXILIARY DISC

0	--	--	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--
50	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--
70	--	--	--	FMG07^	FMG07^	FMG21^	FMG21^	FMGR^	FMGR^	X..07^
80	LG	LG	LG	LG	LG	LG	LG	LG	LG	LG
90	LG	LG	LG	LG	LG	LG	LG	LG	LG	LG
100	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
110	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
120	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
130	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
140	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
150	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
160	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
170	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
180	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
190	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
200	FMP	D.RTR	D.RTR	FMP	FMP	FMP	FMP	FMP	FMP	FMP

THE LS TRACK(S) START AT TRACK 44 OF LU 2

TOTAL AVAILABLE TRACKS = 127  
LARGEST CONTIGUOUS TRACK BLOCK = 73

FREE  
TRACK

DIRECTORY  
TRACK

FILE MANAGER  
TRACK

LOAD-AND-GO  
TRACK

SWAPPED PROGRAMS  
"FMG21" AND "X..07"  
STORED HERE (PROGRAM  
NAME FOLLOWED BY "^")

## NOTE:

The "&" suffix denotes the original loaded copy of a program created by the generator or loader. The "^" suffix denotes a copy of a suspended program which has been swapped by RTE.

# SYSTEM BASE PAGE COMMUNICATION AREA

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
01645	XIDEX	Address of current program's ID extension
01646	XMATA	Address of current program's MAT entry
01647	XI	Address of index register save area
01650	EQTA	FWA of Equipment Table
01651	EQT#	Number of EQT entries
01652	DRT	FWA of Device Reference Table, word 1
01653	LUMAX	Number of logical units in DRT
01654	INTBA	FWA of Interrupt Table
01655	INTLG	Number of Interrupt Table Entries
01656	TAT	FWA of Track Assignment Table
01657	KEYWD	FWA of keyword block
I/O MODULE/DRIVER COMMUNICATION		
01660	EQT1 \	Addresses of first 11 words of current EQT entry (see 01771 for last four words)
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EQT5 \	
01665	EQT6 /	
01666	EQT7	
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11 /	
01673	CHAN	Current DCPC channel number
01674	TEG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY
SYSTEM REQUEST PROCESSOR/EXEC COMMUNICATION		
01676	RQCNT	Number of request parameters -1
01677	RQRTN	Return point address
01700	RQP1 \	Addresses of request parameters (set for a maximum of nine parameters)
01701	RQP2	
01702	RQP3	
01703	RQP4 \	
01704	RQP5 /	
01705	RQP6	
01706	RQP7	
01707	RQP8	
01710	RQP9 /	

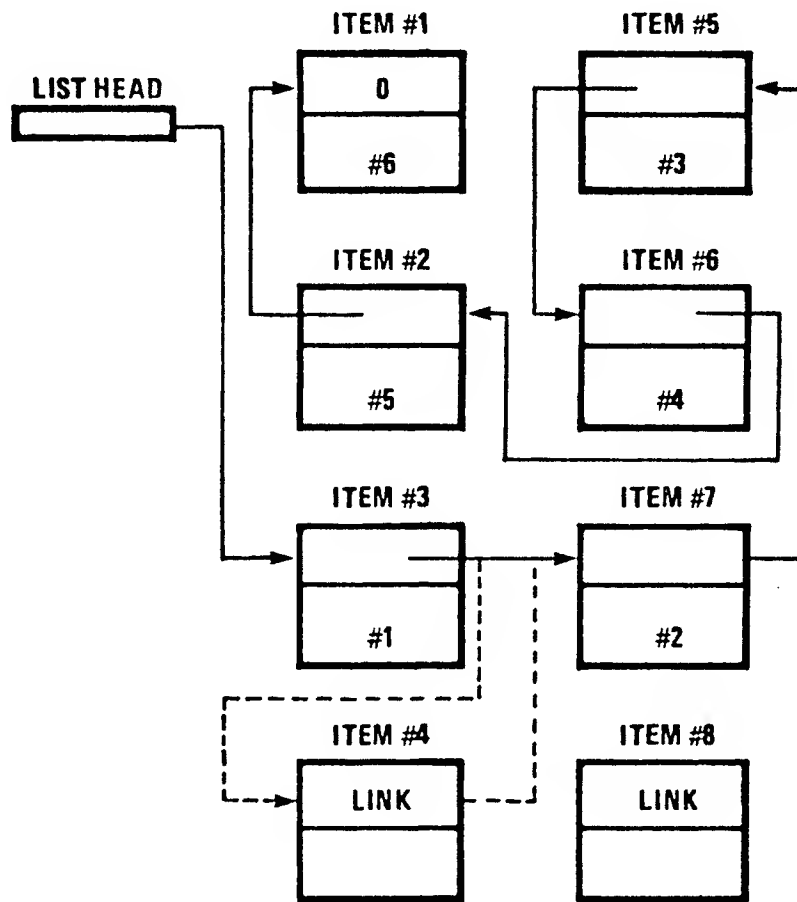
SYSTEM BASE PAGE COMMUNICATION AREA (continued)

Octal Location	Contents	Description
SYSTEM LISTS ADDRESSES		
01711	SKEDD	Schedule list
01713	SUSP2	Wait Suspend list
01714	SUSP3	Available Memory list
01715	SUSP4	Disc Allocation list
01716	SUSP5	Operator Suspend list
PROGRAM ID SEGMENT DEFINITION		
01717	XECT	ID segment address of current program
01720	XLINK	Linkage
01721	XTEMP	Temporary (five words)
01726	XPRI0	Priority word
01727	XPLENT	Primary entry point
01730	XSUSP	Point of suspension
01731	XA	A-register at suspension
01732	XB	B-register at suspension
01733	XEO	E and overflow register suspension
SYSTEM MODULE COMMUNICATION FLAGS		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy interface flag
01740	IDSDA	Disc address of first ID segment
01741	IDSDP	Position within disc sector
MEMORY ALLOCATION BASES DEFINITION		
01742	BPA1	FWA user base page link area
01743	BPA2	LWA user base page link area
01744	BPA3	FWA user base page link
01745	LBORG	FWA of resident library area
01746	RTORG	FWA of real-time COMMON
01747	RTCOM	Length of real-time COMMON
01750 D	RTDRA	FWA of real-time partition
01751 D	AVMEM	LWA+1 of real-time partition
01752	BGORG	FWA of background COMMON
01753	BGCOM	Length of background COMMON
01754 D	BGDRA	FWA of background partition

# SYSTEM BASE PAGE COMMUNICATION AREA (continued)

Octal Location	Contents	Description
UTILITY PARAMETERS		
01755	TATLG	Negative length of track assignment table
01756	TATSD	Number of tracks on system disc
01757	SECT2	Number of sectors/track on LU2 (system)
01760	SECT3	Number of sectors/track on LU3 (aux.)
01761	DSCLB	Disc address of user available library entry points
01762	DSCLN	Number of user available library entry points.
01763	SYSLB	Disc address of system library entry points
01764	SYSLN	Number of system library entry points
01765	LGOTK	LGO: LU#, starting track, number of tracks (same format as ID segment word 28)
01766	LGOC	Current LGO track/sector address (same format as ID segment word 26)
01767	SFCUN	LS: LU# and disc address (same format as ID segment word 26)
01770	MPTFL	Memory protect ON/OFF flag (0/1)
01771	EQT12 \	Address of last four words of current EQT
01772	EQT13 \	
01773	EQT14 /	
01774	EQT15 /	
01775 D	FENCE	Memory protect fence address
01777	BGLWA	LWA memory background partition
D letter indicates the contents of the location are set dynamically by the dispatcher.		

# LINKED LISTS



Linked lists provide a mechanism for quickly ordering and accessing blocks of memory and their constants. It consists of a list head or 'starting point', and a word in each entry pointing to the next entry.

New entries can be included and old entries removed by re-setting only 1 link.

## SYSTEM TABLES

- ID segments, long, short, & extensions
- Equipment Table
- Device Reference Table
- Interrupt Table
- Track Assignment Table
- Class Table
- LU Switch Table
- Resource Number Table
- Keyword Block
- ID Extension Table
- Memory Allocation Table
- Memory Protect Fence Table
- Driver Mapping Table
- Track Map Table

## SYSTEM LISTS

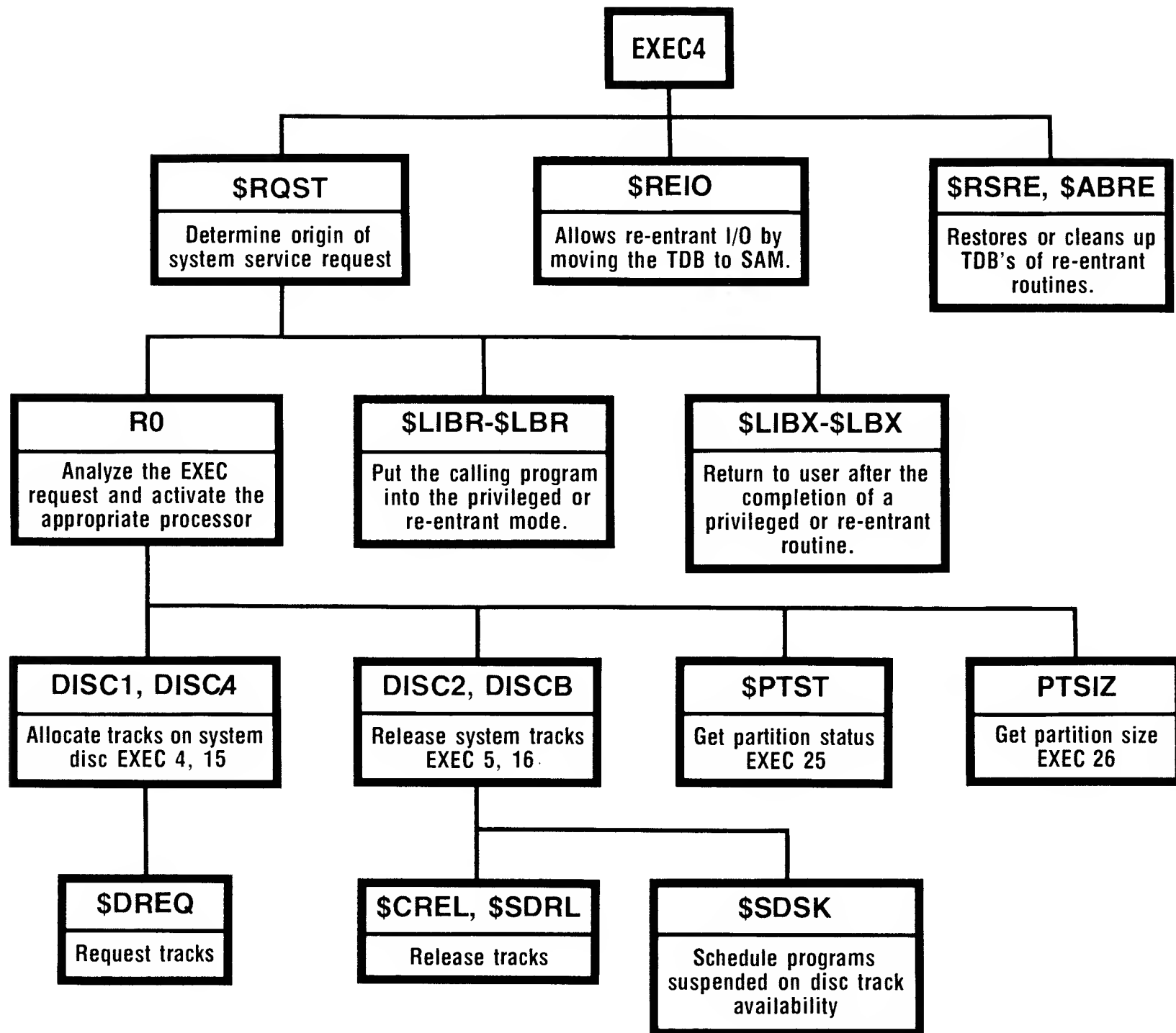
- Schedule List
- General wait List
- Available Memory Suspend List
- Disc Allocation Suspend List
- Operator Suspend List
- I/O Suspend Lists
- Free SAM List

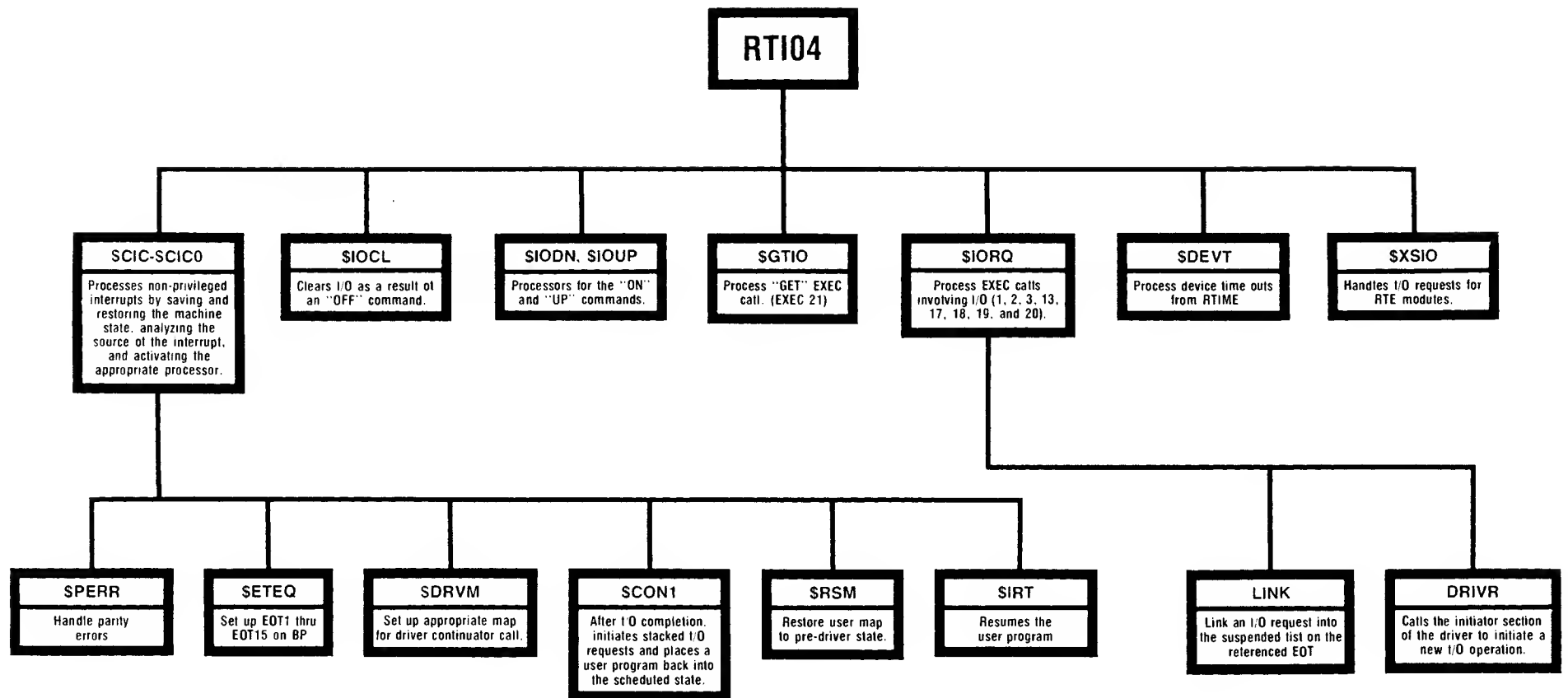
OVERVIEW CHART  
OF RTE TABLES/LISTS

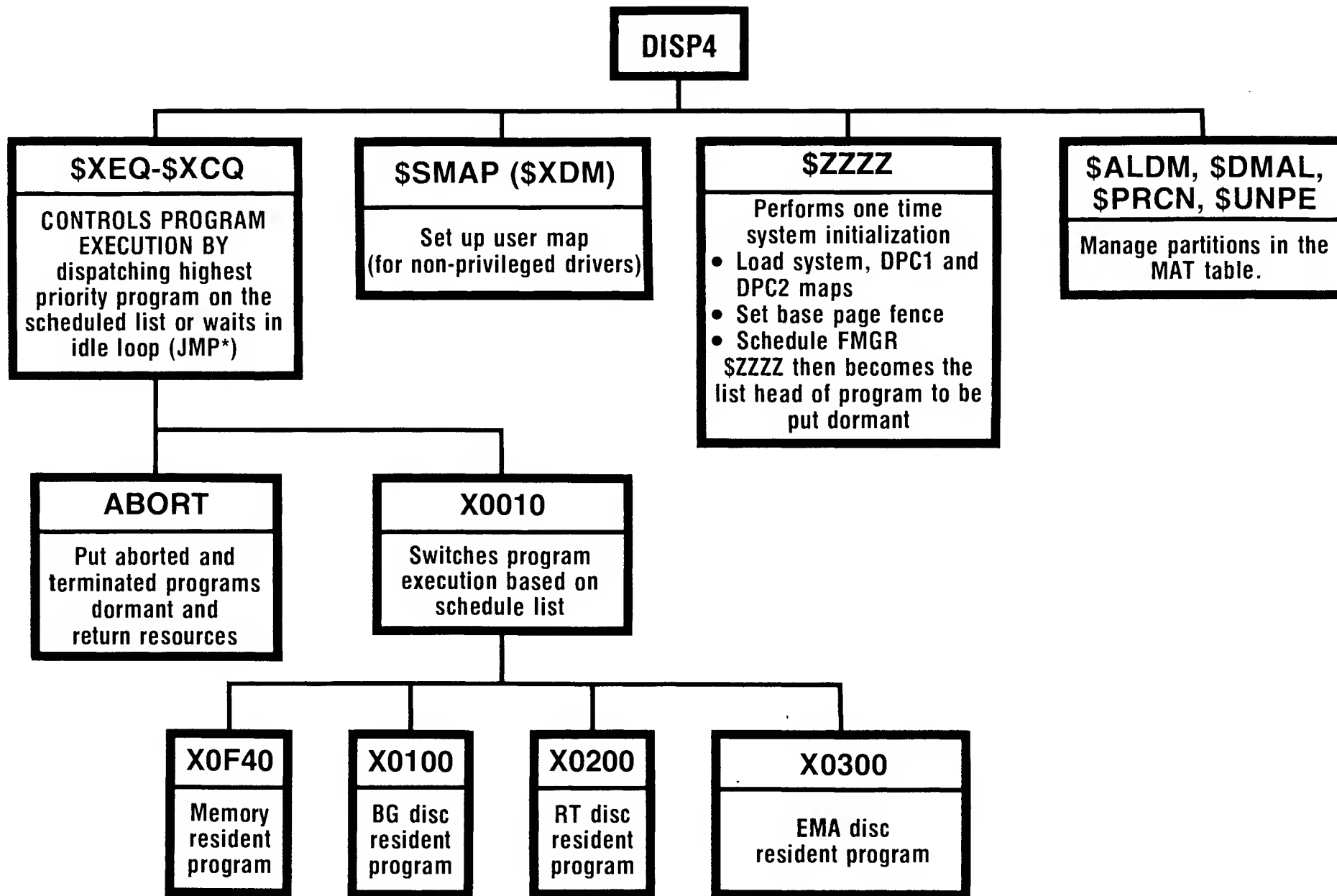


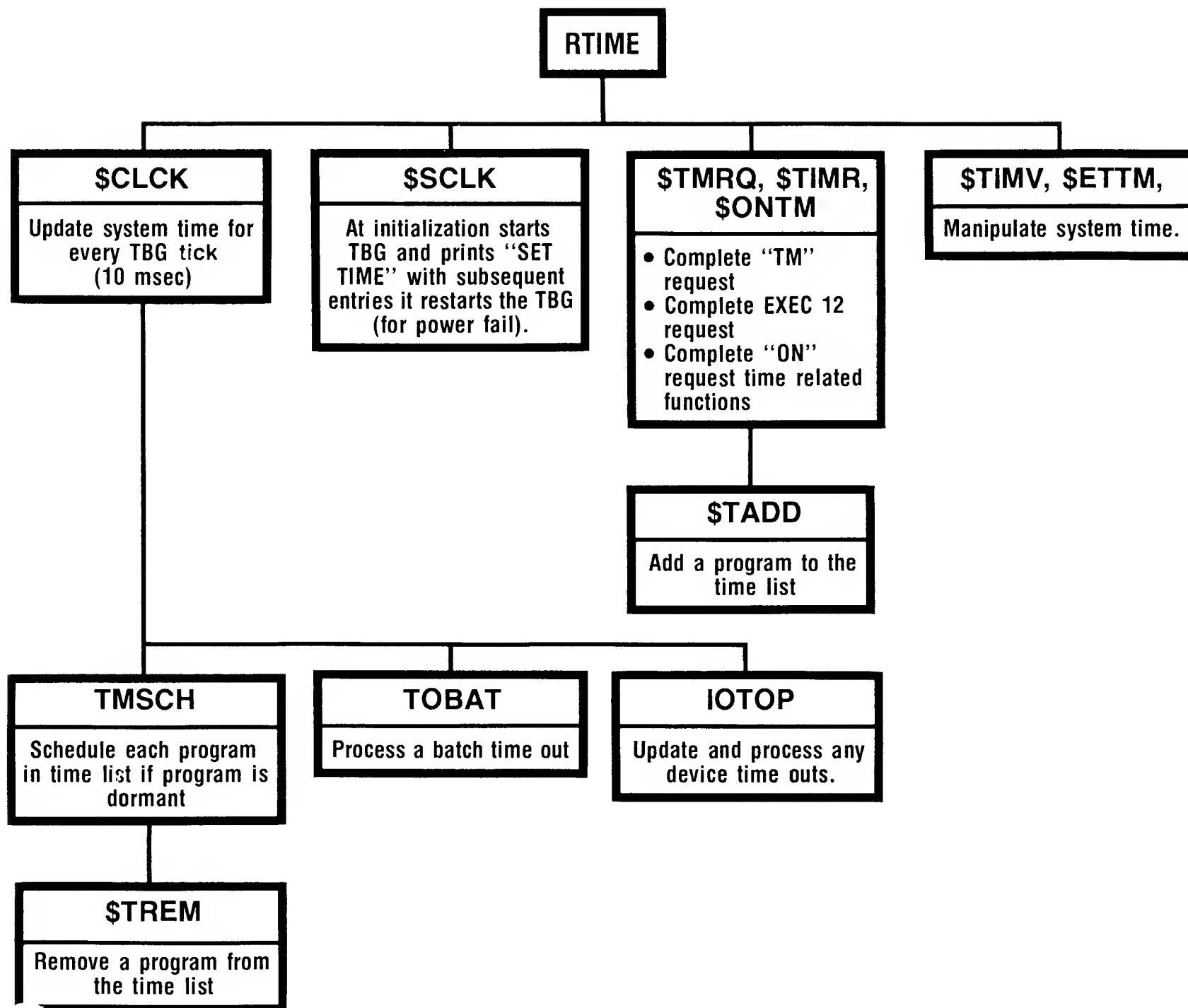
**RTE  
MODULES**

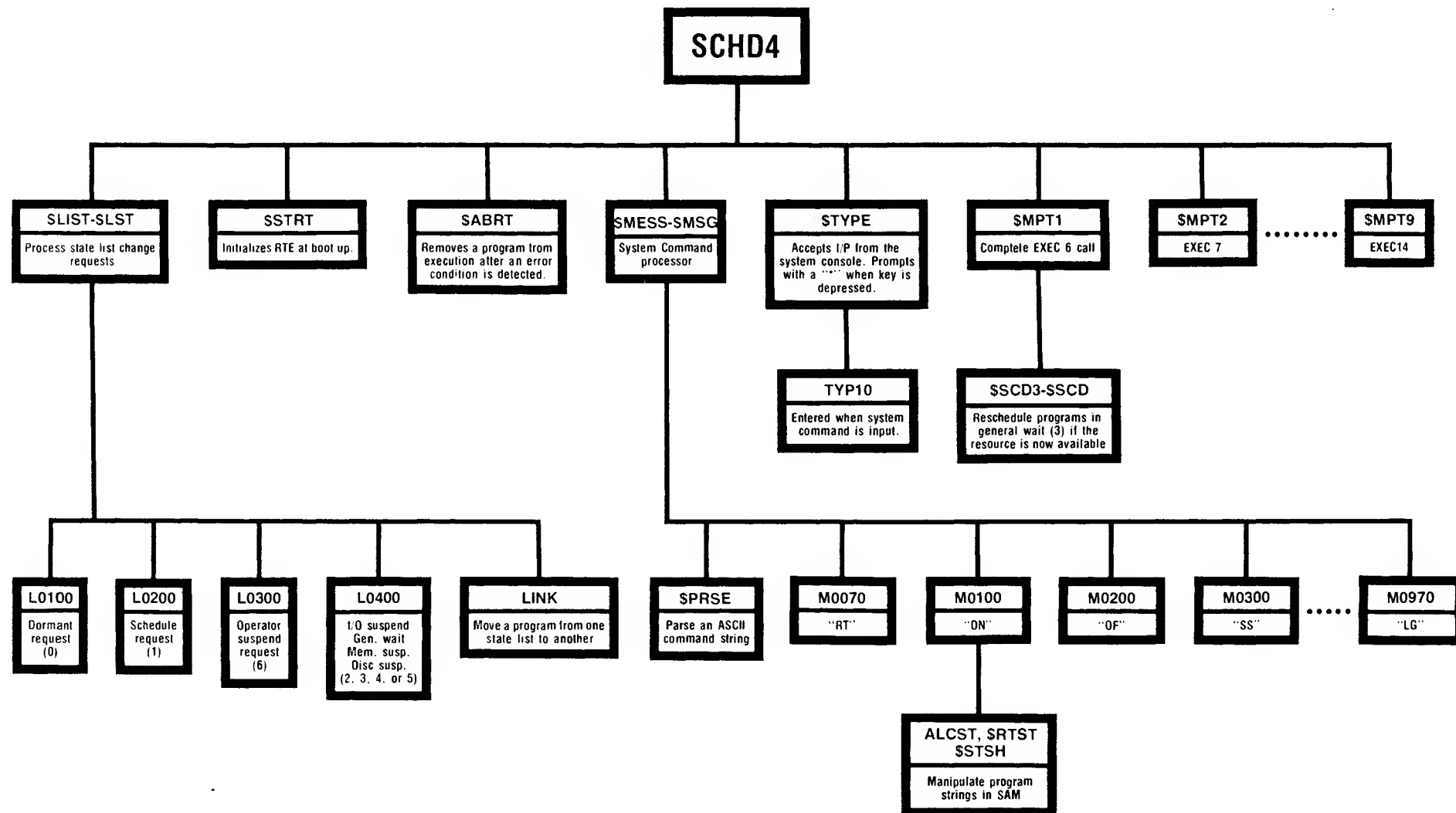












**\$ALC**



```
graph TD; A["$ALC"] --> B["$ALC<br/>Allocate a buffer in SAM"]; A --> C["$RTN<br/>Release a buffer in SAM"]
```

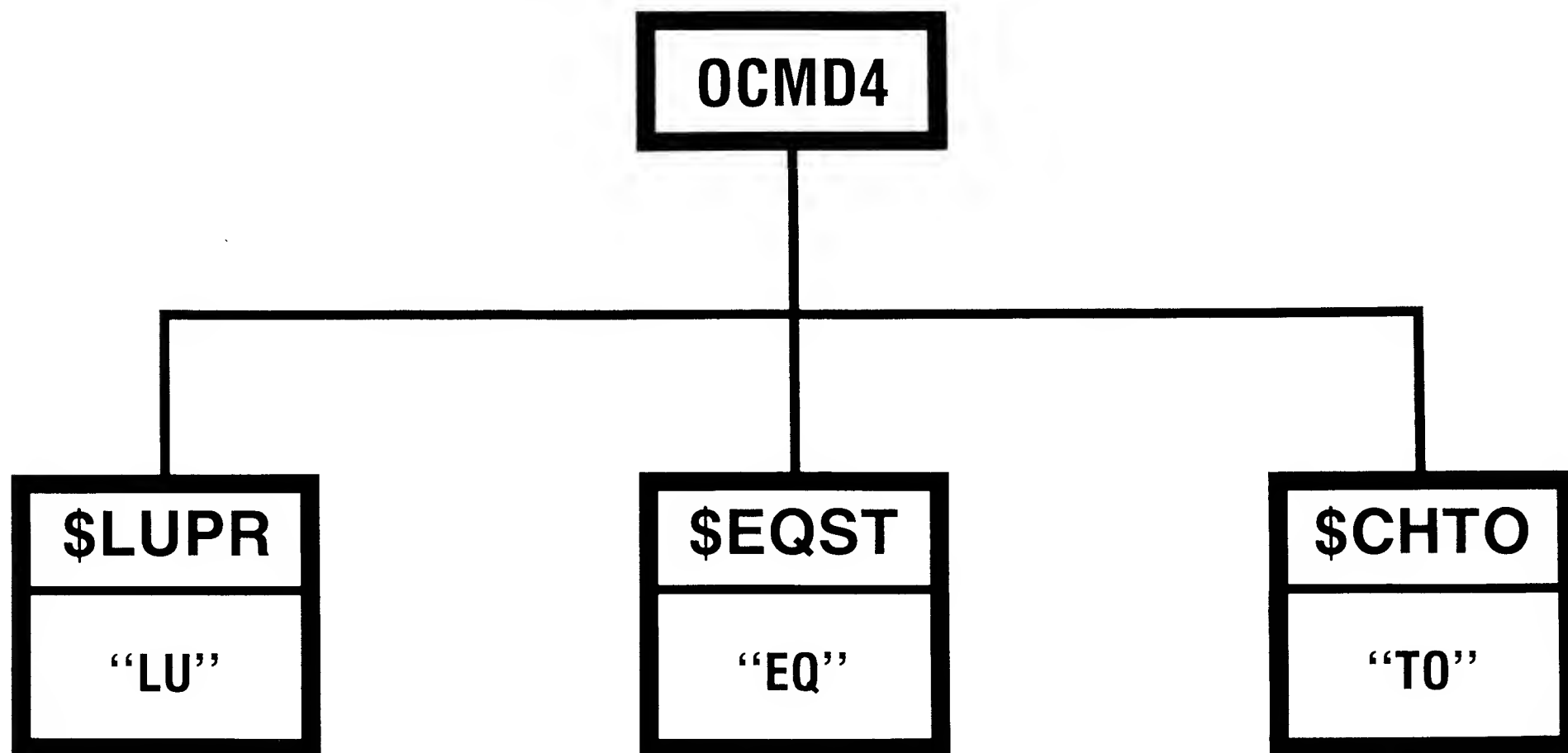
**\$ALC**

**Allocate a buffer in SAM**

**\$RTN**

**Release a buffer in SAM**





## SYSTEM VARIABLES

### I. TABLE AREA I

\$PVCN - Level count for privileged calls  
\$PVST - DMS status for privileged calls  
\$MTM - Class number for PRMPT/\$R\$PN  
\$OPSY - Operating system identification

### II. TABLE AREA II

\$MATA - Address of MAT table  
\$MCHN - Max. mother partition size  
\$MBGP - Max. BG partition size  
\$MRTP - Max. RT partition size  
\$DLTH - No. pages per driver partition  
\$DVPT - Driver ptn. map register number (org 0)  
\$TIME - System time (10's ms (2 words), day/year (1 word))  
\$BATM - Batch timer (2 words 10's ms)  
\$DLP - Logical address of type 4 programs for loading  
\$PLP - Logical address of type 2+3 programs for loading  
\$ENDS - No. of pages in system up to the SAM extension  
\$MPFT - Address of memory protect fence table  
\$BGFR - Listhead of free BG partitions  
\$RTFR - Listhead of free RT partitions  
\$CFR - Listhead of free Mother partitions  
\$IDEX - Address of ID extension table  
\$MRMP - Address of MP map  
\$MPSA - No. of pages and starting page of SAM  
\$MPS2 - No. of pages and starting page of SAM extension  
\$SDA - Starting page of SDA (org. 0)  
\$SDT2 - No. of pages in SDA and Table Area II  
\$CMST - Start page of common (org. 0)  
\$COML - No. of pages in common  
\$MNP - Max. number of partitions  
\$DVMP - Address of driver mapping table  
\$EARP - LWA of MR program area  
\$RL3 - Logical start page of MR library  
\$RLN - No. pages in MR library  
\$SBTE - Disc address of driver partitions #2 onward  
# of pages for driver partitions #2 onward  
Disc address of memory resident base page  
# of pages for memory resident base page  
Disc address of memory resident lib/programs  
# of pages for memory resident lib/programs

24999-16051 1805 SOFTWARE SERVICE KIT SYSTEM  
 MODULE MODULE SIZE (OCTAL) \*\*MXREF\*\*  
 NAME MODULE IDENT. BPAG MAIN COMM

-----  
 FILE NAME: XCR45111 32767: 51 112: 0 IS ON LU 35

SCSY4 92067-16014 REV.1805 780125  
 00000 00000 00000

DISP4 92067-16014 REV.1805 780317  
 00000 05363 00000 ENT= SRENT SBRED SZZZZ SXCO  
 ENT= SALDM SDMAL SSMAP SPRCN  
 ENT= SXDM SMAXP SUNPE  
 EXT= SMRMP SMATA SMPPT SBGFR  
 EXT= SRTFR SEMRP SRSRE SABRT  
 EXT= XSIO SDREQ SWATR STIME  
 EXT= SDREL STRRN SSZIT SABRE  
 EXT= SLIST SRTST SSGAP SERMG  
 EXT= SMCHN SMBGP SMRTP SCFR  
 EXT= SWORK SIOCL SIRT SIDLE  
 EXT= SOVPT SIDEX SCMST SSDA  
 EXT= SSDT2 SMNP SXDMP  
 END= 05214

RTIME 92067-16014 REV.1805 780104  
 00000 00005 00000 ENT= STADD SCLCK STREM STIMV  
 ENT= SETTM STIMR SONTM STMRO  
 ENT= SSCLK  
 EXT= SNER SDEV STLIST SXEQ  
 EXT= SERMG SMSEX SSYMG SIDSM  
 EXT= SWORK SBATH STIME  
 END= 00216

SASC4 92067-16014 REV.1805 780125  
 00000 00073 00000 ENT= SOPER SERIN SNOPG SILST  
 ENT= SNOLG SLGBS SNMEM  
 END= 00050

RTIO4 92067-16014 REV.1805 780310  
 00002 05321 00000 ENT= SCIC0 XSIO SSYMG SIORO  
 ENT= SIOUP SIONN SETEQ SIRT  
 ENT= SDEV SCXC SCYC SGTIO  
 ENT= SUP SCVEQ SDMS SBLLO  
 ENT= SBLUP SBITB SUNLK SXXUP  
 ENT= SOLAY SDMEQ SCKLO SCON1  
 ENT= SCON2 SCON3 SDRVM SRSM  
 ENT= SIOCL  
 EXT= SRQST SCLCK SXEQ STYPE  
 EXT= SLIST SALC SRIN SLUSW  
 EXT= SSCD3 SRNTB SCNV3 SERMG  
 EXT= SCNV1 SCLAS SREIO SABRT  
 EXT= SNER SZZZZ SPDSK SUCON  
 EXT= SUIN SCIC SPERR SERAB  
 EXT= SIDNO SSMAP SMATA SMRMP  
 EXT= SMVBF SOVPT SDLTH SOVMP  
 EXT= SSDA  
 END= 00000

EXEC4 92067-16014 REV.1805 780310  
 00000 02357 00000 ENT= SERMG SRQST SOTRL SUBER

ENT= \$DREQ \$DREL \$SDRL \$SDSK  
 ENT= \$ERRA \$REIO \$CREL \$RSRE  
 ENT= \$ABRE \$PDSK \$ABXY \$PWR5  
 ENT= \$MVBFB \$SGAF \$LEND \$DHED  
 ENT= \$LBR \$LBX \$XEX  
 EXT= \$CNV3 \$SYMG \$LIST \$XEQ  
 EXT= \$PVCN EXEC \$LIBR \$LIBX  
 EXT= \$IDLE \$PVST \$RENT \$CVEQ  
 EXT= \$ABRT \$DMS \$TRRN \$SCLK  
 EXT= \$ALC \$RTN \$MATA \$IDNO  
 EXT= \$MRMP \$PBUF \$MNP \$MPFT  
 EXT= \$PERR \$CNV1 \$IORQ \$MPT1  
 EXT= \$MPT2 \$MPT3 \$MPT4 \$MPT5  
 EXT= \$MPT6 \$MPT7 \$MPT9 \$GTIO  
 EXT= \$MPT8  
 END= 00000

FILE NAME: XCR4S2:: 32767: 5: 106: 0 IS ON LU 35

STRN4 92067-16014 REV.1805 780104  
 00000 00153 00000

ENT= \$TRRN \$CRN\* \$ULU  
 EXT= \$RNTB \$IDNO \$SCD3 \$SCLK  
 EXT= \$ULLU \$CGRN  
 END= 00000

SCHD4 92067-16014 REV.1805 780317  
 00000 05160 00000

ENT= \$ABRT \$TYPE \$PRSE \$CNV1  
 ENT= \$CNV3 \$OP \$MPT1 \$MPT2  
 ENT= \$MPT3 \$MPT4 \$MPT5 \$MPT6  
 ENT= \$STRT \$INER \$MPT7 \$ASTM  
 ENT= \$WATR \$SZIT \$MPT8 \$IDSM  
 ENT= \$PBUF \$MPT9 \$RTST \$CVWD  
 ENT= \$STRG \$MSEX \$LSTM \$LST  
 ENT= \$SCD \$ID# \$MSG  
 EXT= \$XSIO \$IOUP \$IODN \$ERMG  
 EXT= \$DREQ \$DLP \$PLP \$MPFT  
 EXT= \$MEU \$CMST \$COML \$SDA  
 EXT= \$SDT2 \$RLB \$RLN \$MPSA  
 EXT= \$MPS2 \$IDEX \$IOCL \$OTRL  
 EXT= \$DREL \$CHTO \$LUPR \$EQST  
 EXT= \$MESS \$LIST \$IDNO \$SCD3  
 EXT= \$CNFG \$ERAB \$ZZZZ \$TIME  
 EXT= \$PVCN \$MNP \$ERIN \$NOPG  
 EXT= \$OPER \$ILST \$NOLG \$LGBS  
 EXT= \$NMEM \$XEQ \$TMRQ \$ONTM  
 EXT= \$ALC \$RTN \$WORK \$BRED  
 EXT= \$TIMR \$ETTM \$TIMV \$TREM  
 EXT= \$RNTB \$CREL \$SYMG \$SDRL  
 EXT= \$ALDM \$DMAL \$MATA \$PRCN  
 EXT= \$MBGP \$MRTP \$MCHN \$MAXP  
 EXT= \$BLL0 \$BLUP  
 END= 00047

\$ALC 92067-16014 REV.1805 741120  
 00000 00206 00000

ENT= \$ALC \$RTN \$PNTR  
 EXT= \$LIST \$WORK  
 END= 00000

OCMD4 92067-16014 REV.1805 771102  
 00000 01142 00000

ENT= \$LUPR \$EQST \$CHTO  
 EXT= \$CVEQ \$CNV1 \$CNV3 \$UNLK

EXT# SXXUP SDLAY SDMEQ SSCD3  
 EXT# SETEQ SCKLO SBITB SINER  
 EXT# SXCQ SMSEX  
 END# 00000

PERR4

92067-16014 REV.1805 780227

00001 00741 00000

ENT# SPERR SPETB  
 EXT# SCNV1 SCNV3 SSYMG SERMG  
 EXT# SXCQ SUNPE SMAXP SMATA  
 EXT# SDMS SABXY SCIC  
 END# 00000

SCNFG

92067-16014 REV.1805 770112

00000 04637 00000

ENT# SCNFG SEXIT SPCHN SWRRD  
 ENT# SUSRS SABDP SSMTB STRTB  
 ENT# STREN SNPGQ SGDPG SSAVE  
 EXT# SSBTB SXSID SCMST SENDS  
 EXT# SMRMP SXCQ SLIST SCNV3  
 EXT# SPRSE SPLP SMATA SMNP  
 END# 00312

SSTB1

92067-16014 REV.1805 780223

00000 00123 00000

ENT# SERAB SPVCN EXEC SLIBR  
 ENT# SLIBX SPVST SUPIO SCIC  
 ENT# SXCIC SYCIC SUIN SUCON  
 ENT# SXEQ SXOMP SIDLE SSCD3  
 ENT# SIDNO SMEU SLIST SMESS  
 ENT# SWORK SSOP SULLU SCGRN  
 ENT# SMTM SOPSY  
 EXT# SERRA SLBR SLBX SXEX  
 EXT# SUP SCIC0 SCXC SCYC  
 EXT# SCON1 SCON2 SCON3 SXCQ  
 EXT# SXDM SSCD SIDW SLST  
 EXT# SMSG SIDSM SOP SULU  
 EXT# SCRNM  
 END# 00122

SSTB2

92067-16014 REV.1805 771107

00000 00045 00000

ENT# SMATA SMCHN SMBGP SMRTP  
 ENT# SDLTH SDVPT STIME SBATH  
 ENT# SOLP SPLP SENDS SMPFT  
 ENT# SBGFR SRTFR SIDEX SMRMP  
 ENT# SMP32 SEMRP SMP3A SSDA  
 ENT# SSDT2 SCMST SCOML SCFR  
 ENT# SMNP SDVMP SRLB SRLN  
 ENT# SSBTB  
 END# 00000

TOTAL 00003 33056 00000

MODULE LEVEL MODULES WHERE USED

MODULE	LEVEL	MODULES WHERE USED
\$STB1	100	\$TRN4 RTIO4 PERR4 SCHD4 DISP4 EXEC4 RTIME \$ALC \$CNFG OCMD4
\$STB2	100	RTIME DISP4 SCHD4 \$CNFG RTIO4 EXEC4 PERR4
\$ALC	100	RTIO4 EXEC4 SCHD4
\$ASC4	100	SCHD4
\$CNFG	100	SCHD4
\$CSY4	1	
\$TRN4	100	\$STB1 DISP4 EXEC4
DISP4	100	SCHD4 PERR4 EXEC4 RTIO4 OCMD4 \$CNFG \$STB1
EXEC4	100	DISP4 PERR4 SCHD4 RTIME RTIO4 \$STB1
OCMD4	100	SCHD4
PERR4	100	RTIO4 EXEC4
RTIME	100	RTIO4 SCHD4 EXEC4 \$TRN4
RTIO4	100	OCMD4 SCHD4 \$STB1 EXEC4 RTIME PERR4 DISP4 \$CNFG
SCHD4	100	DISP4 RTIO4 EXEC4 OCMD4 PERR4 \$CNFG \$STB1 RTIME

ENTRY DEFN-MOD MODULES WHERE USED

SSOP	SSTB1	
SABOP	SCNFG	
SABRE	EXEC4	DISP4
SABRT	SCHD4	DISP4 RTIO4 EXEC4
SABXY	EXEC4	PERR4
SALC	SALC	RTIO4 EXEC4 SCHD4
SALOM	DISP4	SCHD4
SASTM	SCHD4	
SBATH	SSTB2	RTIME
SBGFR	SSTB2	DISP4
SBITB	RTIO4	OCMD4
SBLLLO	RTIO4	SCHD4
SBLUP	RTIO4	SCHD4
SBRED	DISP4	SCHD4
SCFR	SSTB2	DISP4
SCGRN	SSTB1	STRN4
SCHTO	OCMD4	SCHD4
SCIC	SSTB1	RTIO4 PERR4
SCIC0	RTIO4	SSTB1
SCKLO	RTIO4	OCMD4
SCLCK	RTIME	RTIO4
SCMST	SSTB2	DISP4 SCHD4 SCNFG
SCNFG	SCNFG	SCHD4
SCNV1	SCHD4	RTIO4 EXEC4 OCMD4 PERR4
SCNV3	SCHD4	RTIO4 EXEC4 OCMD4 PERR4 SCNFG
SCOML	SSTB2	SCHD4
SCON1	RTIO4	SSTB1
SCON2	RTIO4	SSTB1
SCON3	RTIO4	SSTB1
SCREL	EXEC4	SCHD4
SCRN#	STRN4	SSTB1
SCVEQ	RTIO4	EXEC4 OCMD4
SCVWD	SCHD4	
SCXC	RTIO4	SSTB1
SCYC	RTIO4	SSTB1
SDEV	RTIO4	RTIME
SDHED	EXEC4	
SDLAY	RTIO4	OCMD4
SDLP	SSTB2	SCHD4
SDLTH	SSTB2	RTIO4
SDMAL	DISP4	SCHD4
SDMEQ	RTIO4	OCMD4
SDMS	RTIO4	EXEC4 PERR4
SDREL	EXEC4	DISP4 SCHD4
SDREQ	EXEC4	DISP4 SCHD4
SDRVM	RTIO4	
SDVMP	SSTB2	RTIO4
SDVPT	SSTB2	DISP4 RTIO4
SEMHP	SSTB2	DISP4
SEND5	SSTB2	SCNFG
SEQST	OCMD4	SCHD4
SERAB	SSTB1	RTIO4 SCHD4
SERIN	SASC4	SCHD4
SERMG	EXEC4	DISP4 RTIME RTIO4 SCHD4 PERR4
SERRA	EXEC4	SSTB1
SETEQ	RTIO4	OCMD4
SETTM	RTIME	SCHD4
SEXIT	SCNFG	

\$GDPG	\$CNFG	
\$GTIO	RTIO4	EXEC4
\$ID#	SCHD4	\$STB1
\$IDEX	\$STB2	DISP4 SCHD4
\$IDLE	\$STB1	DISP4 EXEC4
\$IDNO	\$STB1	RTIO4 EXEC4 \$TRN4 SCHD4
\$IDSM	SCHD4	RTIME \$STB1
\$ILST	\$ASC4	SCHD4
\$INER	SCHD4	RTIME RTIO4 OCMD4
\$IOCL	RTIO4	DISP4 SCHD4
\$IODN	RTIO4	SCHD4
\$IORQ	RTIO4	EXEC4
\$IOUP	RTIO4	SCHD4
\$IRT	RTIO4	DISP4
\$LBR	EXEC4	\$STB1
\$LBX	EXEC4	\$STB1
\$LEND	EXEC4	
\$LGBS	\$ASC4	SCHD4
\$LIBR	\$STB1	EXEC4
\$LIBX	\$STB1	EXEC4
\$LIST	\$STB1	DISP4 RTIME RTIO4 EXEC4 SCHD4 \$ALC \$CNFG
\$LST	SCHD4	\$STB1
\$LSTM	SCHD4	
\$LUPR	OCMD4	SCHD4
\$MATA	\$STB2	DISP4 RTIO4 EXEC4 SCHD4 PERR4 \$CNFG
\$MAXP	DISP4	SCHD4 PERR4
\$MBGP	\$STB2	DISP4 SCHD4
\$MCHN	\$STB2	DISP4 SCHD4
\$MESS	\$STB1	SCHD4
\$MEU	\$STB1	SCHD4
\$MNP	\$STB2	DISP4 EXEC4 SCHD4 \$CNFG
\$MPFT	\$STB2	DISP4 EXEC4 SCHD4
\$MPS2	\$STB2	SCHD4
\$MPSA	\$STB2	SCHD4
\$MPT1	SCHD4	EXEC4
\$MPT2	SCHD4	EXEC4
\$MPT3	SCHD4	EXEC4
\$MPT4	SCHD4	EXEC4
\$MPT5	SCHD4	EXEC4
\$MPT6	SCHD4	EXEC4
\$MPT7	SCHD4	EXEC4
\$MPT8	SCHD4	EXEC4
\$MPT9	SCHD4	EXEC4
\$MRMP	\$STB2	DISP4 RTIO4 EXEC4 \$CNFG
\$MRTP	\$STB2	DISP4 SCHD4
\$MSEX	SCHD4	RTIME OCMD4
\$MSG	SCHD4	\$STB1
\$MTM	\$STB1	
\$MVBF	EXEC4	RTIO4
\$NMEM	\$ASC4	SCHD4
\$NOLG	\$ASC4	SCHD4
\$NOPG	\$ASC4	SCHD4
\$NPGQ	\$CNFG	
\$ONTM	RTIME	SCHD4
\$OP	SCHD4	\$STB1
\$UPER	\$ASC4	SCHD4
\$OPSY	\$STB1	
\$OTRL	EXEC4	SCHD4
\$PBUF	SCHD4	EXEC4
\$PCHN	\$CNFG	



SPDSK	EXEC4	RTIO4
SPERR	PERR4	RTIO4 EXEC4
SPETB	PERR4	
SPLP	SSTB2	SCHD4 SCNFG
SPNTR	SALC	
SPRCN	DISP4	SCHD4
SPRSE	SCHD4	SCNFG
SPVCN	SSTB1	EXEC4 SCHD4
SPVST	SSTB1	EXEC4
SPWR5	EXEC4	
SREIO	EXEC4	RTIO4
SRENT	DISP4	EXEC4
SRLB	SSTB2	SCHD4
SRLN	SSTB2	SCHD4
SRQST	EXEC4	RTIO4
SRSM	RTIO4	
SRSRE	EXEC4	DISP4
SRTFR	SSTB2	DISP4
SRTN	SALC	RTIO4 EXEC4 SCHD4
SRTST	SCHD4	DISP4
SSAVE	SCNFG	
SSBTB	SSTB2	SCNFG
SSCD	SCHD4	SSTB1
SSCD3	SSTB1	RTIO4 STRN4 SCHD4 OCMD4
SSCLK	RTIME	EXEC4 STRN4
SSDA	SSTB2	DISP4 RTIO4 SCHD4
SSDRL	EXEC4	SCHD4
SSDSK	EXEC4	
SSDT2	SSTB2	DISP4 SCHD4
SSGAF	EXEC4	DISP4
SSMAP	DISP4	RTIO4
SSMTB	SCNFG	
SSTRG	SCHD4	
SSTRT	SCHD4	
SSYMG	RTIO4	RTIME EXEC4 SCHD4 PERR4
SSZIT	SCHD4	DISP4
STADD	RTIME	
STIME	SSTB2	DISP4 RTIME SCHD4
STIMR	RTIME	SCHD4
STIMV	RTIME	SCHD4
STMRQ	RTIME	SCHD4
STREM	RTIME	SCHD4
STREN	SCNFG	
STRRN	STRN4	DISP4 EXEC4
STRTB	SCNFG	
STYPE	SCHD4	RTIO4
SUCON	SSTB1	RTIO4
SUIN	SSTB1	RTIO4
SULLU	SSTB1	STRN4
SULU	STRN4	SSTB1
SUNLK	RTIO4	OCMD4
SUNPE	DISP4	PERR4
SUP	RTIO4	SSTB1
SUPIO	SSTB1	
SUSER	EXEC4	
SUSRS	SCNFG	
SWATR	SCHD4	DISP4
SWORK	SSTB1	DISP4 RTIME SCHD4 SALC
SWRRD	SCNFG	
SXCIC	SSTB1	

\$XCQ	DISP4	OCMD4 PERR4 \$CNFG \$STB1
\$XDM	DISP4	\$STB1
\$XDMP	\$STB1	DISP4
\$XEQ	\$STB1	RTIME RTIO4 EXEC4 SCHD4
\$XEX	EXEC4	\$STB1
\$XSIO	RTIO4	DISP4 SCHD4 \$CNFG
\$XXUP	RTIO4	OCMD4
\$YCIC	\$STB1	
\$ZZZZ	DISP4	RTIO4 SCHD4
EXEC	\$STB1	EXEC4

UNRESOLVED EXT MODULES WHERE USED

-----  
\$CLAS RTI04  
\$LUSH RTI04  
\$RNTB RTI04 \$TRN4 \$CHD4  
END OF CROSS REF

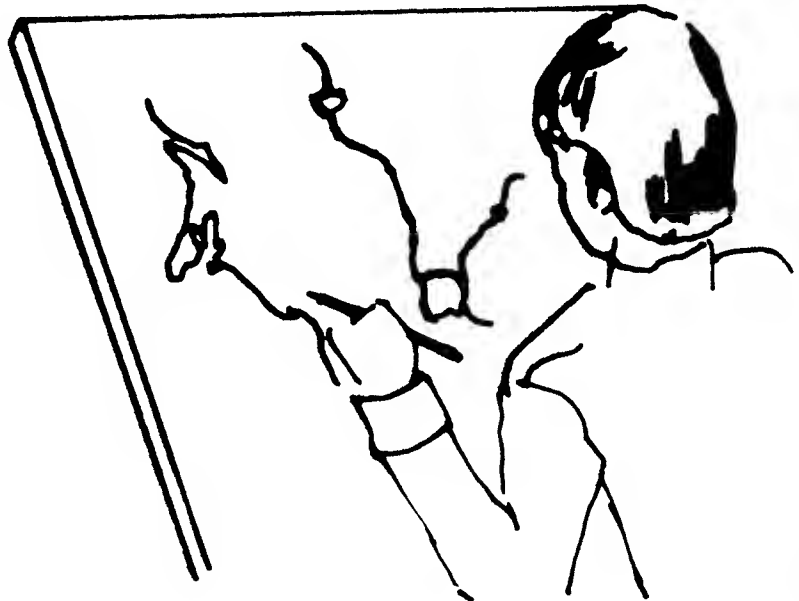
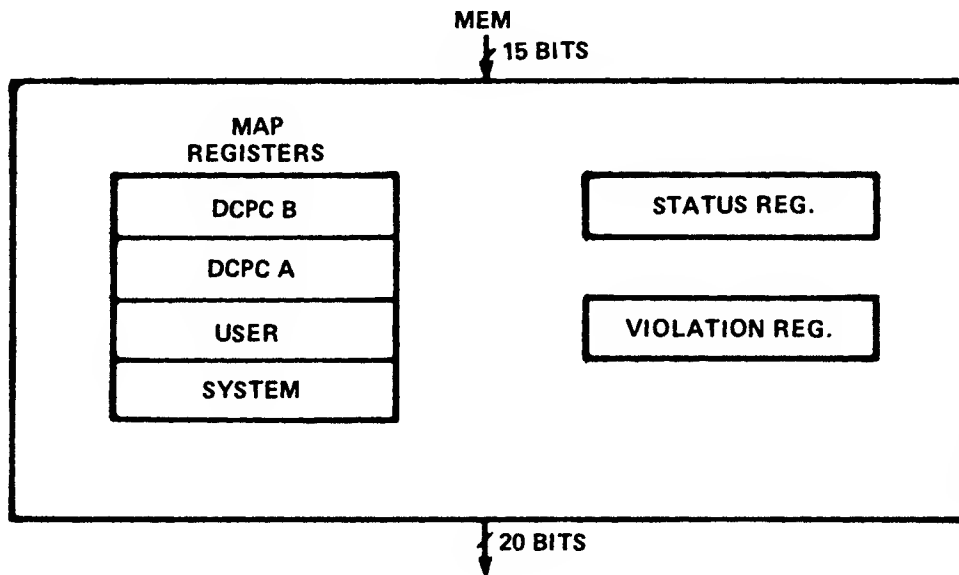


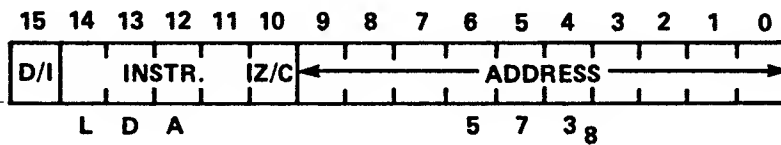
# ***DYNAMIC MAPPING***



# DYNAMIC MAPPING SYSTEM (DMS)

- DMS CONSISTS OF:
  - 1) MEMORY PROTECT (MP)
  - 2) DMS INSTRUCTIONS (ROM)
  - 3) MEMORY EXPANSION MODULE (MEM)



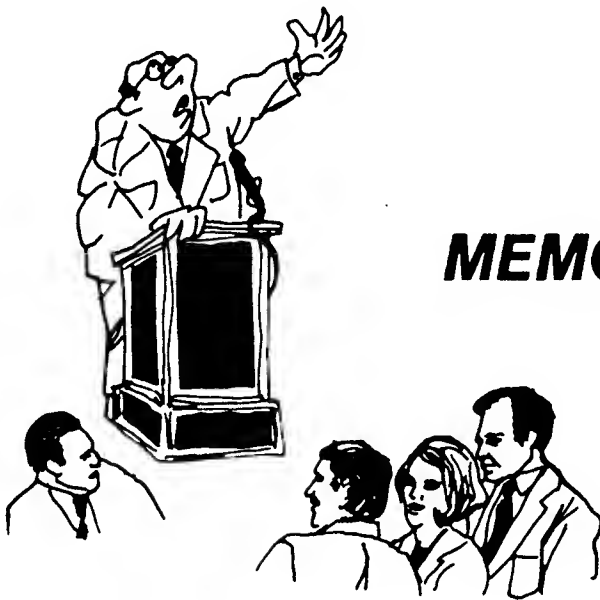


– MEMORY ADDRESSING REQUIRES 15 BITS!

– THE INSTRUCTION PROVIDES 10

—→ WHERE DO THE OTHER 5 COME FROM?

- a) IF BIT 15 IS SET (INDIRECT ADDRESSING), THEY'RE TAKEN FROM THE FINAL ADDRESS. OTHERWISE,
- b) IF BIT 10 IS CLEAR, THEY'RE SET = 0
- c) IF BIT 10 IS SET, THEY'RE SET = TO THE UPPER 5 BITS SPECIFYING THE ADDRESS OF THE INSTRUCTION ("P" REGISTER)



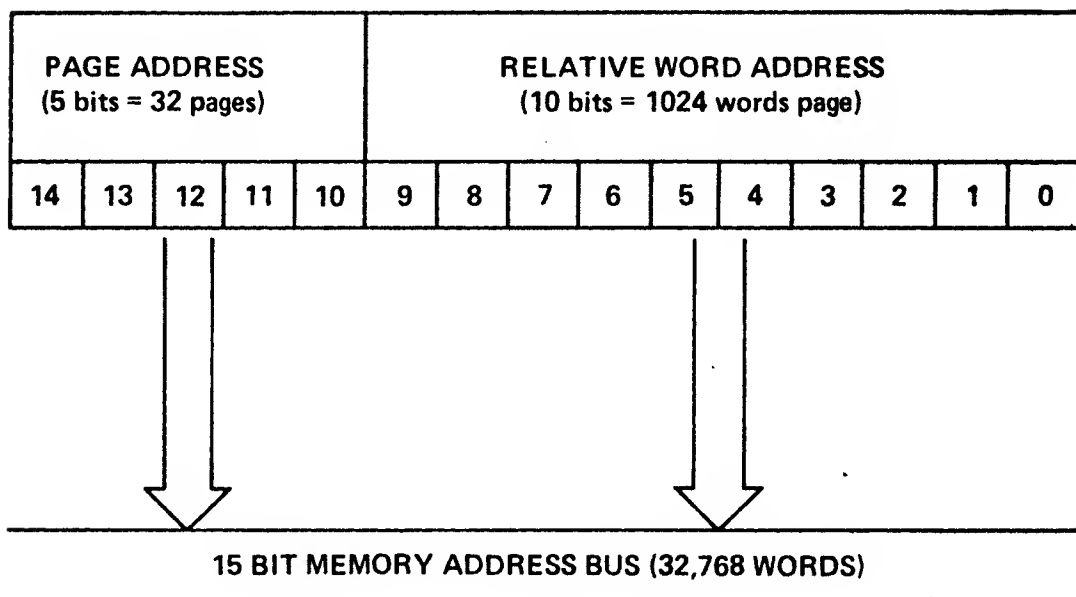
## MEMORY ADDRESSING



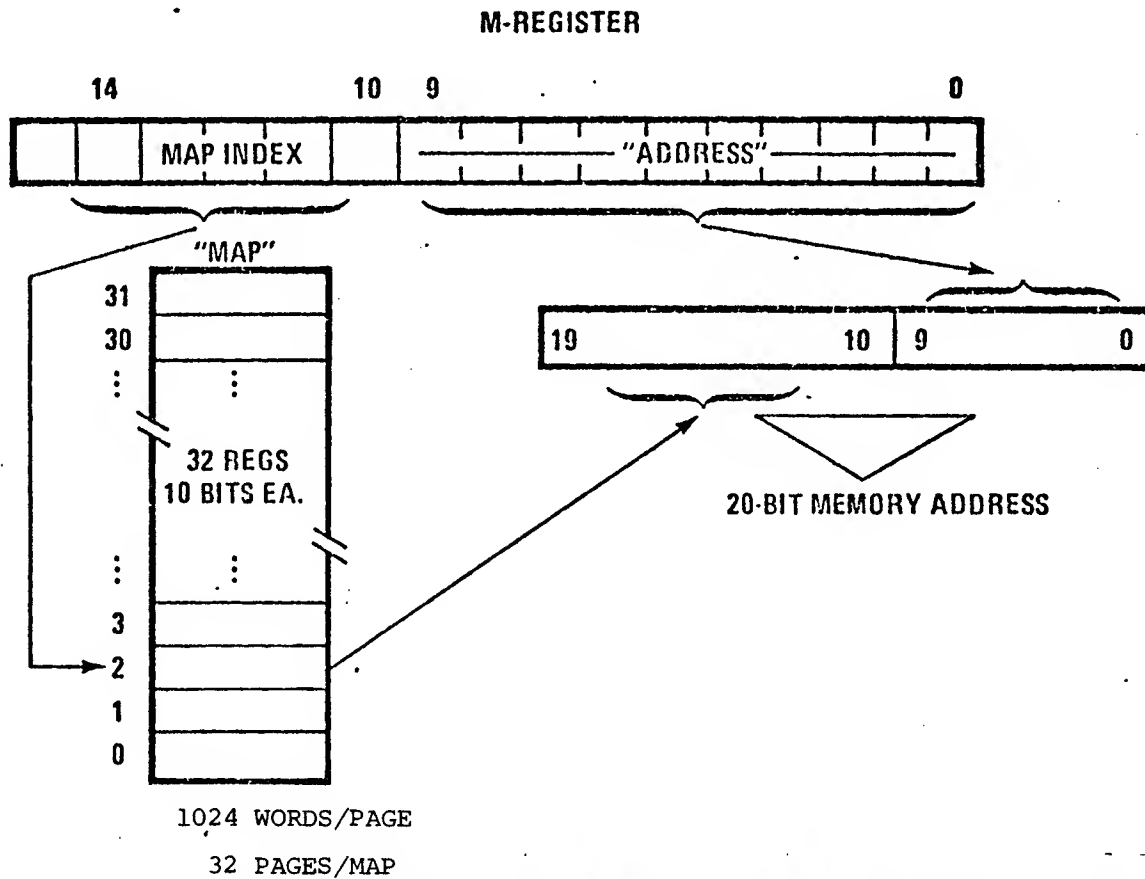
# **\* BASIC 21MX**

## **ADDRESSES 32K WORDS**

**M-REGISTER (15 BITS)**



21MX WITH DMS ADDRESSES UP TO 1 MEGA WORD



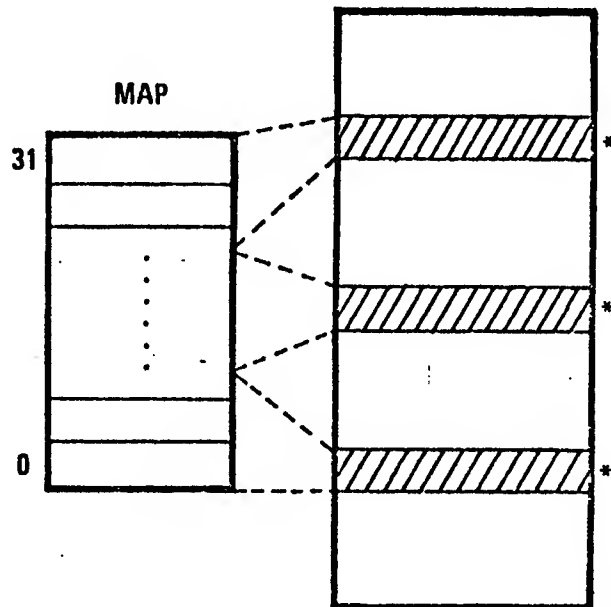
# PHYSICAL MEMORY

## NOTE:

THE 5-BIT "MAP INDEX" CAN ONLY SPECIFY UP TO 32 REGISTERS. THUS, WITH A GIVEN MAP WE CAN STILL ONLY ADDRESS (ACCESS) 32 PAGES OF MEMORY.

THIS 32K SUB-SET IS OUR "LOGICAL ADDRESS SPACE" OR "LOGICAL MEMORY"

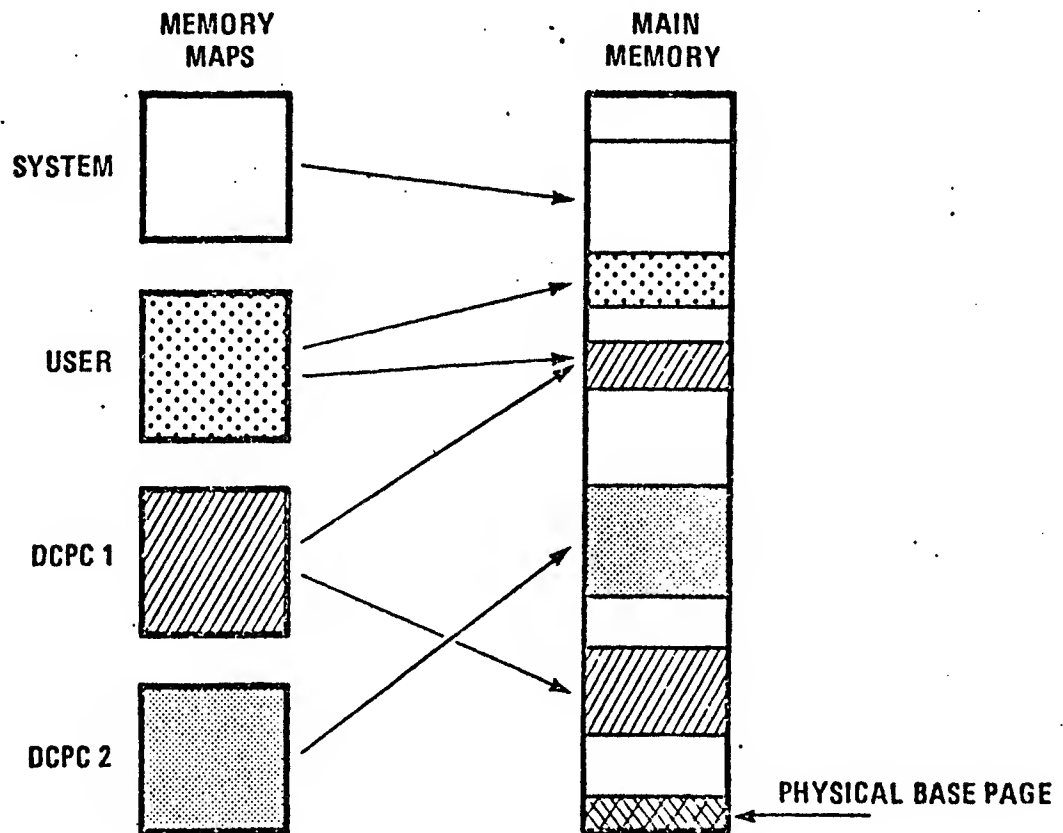
\*32 PAGES TOTAL



## 21 MX DYNAMIC MAPPING

- a) TRANSPARENT TO USER INSTRUCTIONS
- b) A HARDWARE/FIRMWARE SCHEME
- c) 12-BIT MAP REGISTERS ALLOW READ/WRITE PROTECTION
- d) STATUS AND VIOLATION REGISTERS
- e) USES 4 INDEPENDENT MAPS
- f) MAP CONTENTS ARE PROGRAMMABLE (SYA, USA, etc.)
- g) MAPPING CAN BE ENABLED/DISABLES
- h) SPECIAL INSTRUCTIONS ALLOW "CROSS-MAP-MOVES" BETWEEN SYSTEM AND USER MAPS (XLA, XSA, etc.)
- i) PROGRAMMABLE BASE PAGE FENCE (LFA, etc.)

# MAPS DYNAMICALLY CONFIGURE MEMORY

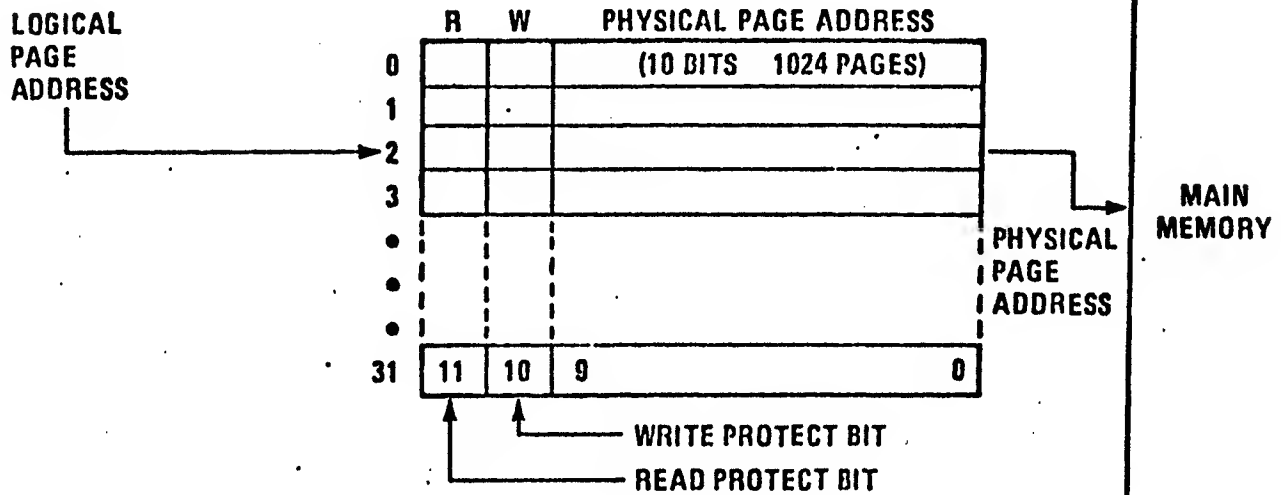


ALL MAPS INCLUDE PART OF THE PHYSICAL BASE PAGE

## MAP SEGMENTATION

171 <sub>8</sub>	PORT B MAP (32 REGISTERS)	127 <sub>10</sub>
140 137	PORT A MAP (32 REGISTERS)	96 95
100 77	USER MAP (32 REGISTERS)	64 63
40 37	SYSTEM MAP (32 REGISTERS)	32 31
0		0

# MAP REGISTER CONTENTS:



UNUSED MAP REGISTERS HAVE BOTH "R" AND "W" BITS SET

# DMS (MEM) REGISTERS

MEM Status Register Format

BITS	SIGNIFICANCE
15	0 = MEM disabled at last interrupt 1 = MEM enabled at last interrupt
14	0 = System map selected at last interrupt 1 = User map selected at last interrupt
13	0 = MEM disabled currently 1 = MEM enabled currently
12	0 = System map selected currently 1 = User map selected currently
11	0 = Protected mode disabled currently 1 = Protected mode enabled currently
10	Portion mapped*
9	Base page fence bit 9
8	Base page fence bit 8
7	Base page fence bit 7
6	Base page fence bit 6
5	Base page fence bit 5
4	Base page fence bit 4
3	Base page fence bit 3
2	Base page fence bit 2
1	Base page fence bit 1
0	Base page fence bit 0

*Bit 10	Mapped Address (M)
0	Fence $\leq M < 2000_8$
1	$1 < M < \text{Fence}$

Note: The base page fence separates the reserved (mapped) memory from the shared (un-mapped) memory. Bit 10 specifies which area is reserved (mapped).

MEM Violation Register Format

BITS	SIGNIFICANCE
15	Read violation*
14	Write violation*
13	Base page violation*
12	Privileged instruction violation*
11	Reserved
10	Reserved
9	Reserved
8	Reserved
7	0 = ME bus disabled at violation 1 = ME bus enabled at violation
6	0 = MEM disabled at violation 1 = MEM enabled at violation
5	0 = System map enabled at violation 1 = User map enabled at violation
4	Map address bit 4
3	Map address bit 3
2	Map address bit 2
1	Map address bit 1
0	Map address bit 0

\*Significant when associated bit is set.

If MP is enabled any of these 4 violations will cause an interrupt of SC 5.

These registers are read with RSA/B or RVA/B instructions.



## DMS ERRORS

DMS errors generate an interrupt on SC5 along with MP errors and memory parity errors. DMS errors are caused by:

- Reading a read protected page.
- Writing into a write protected page
- Base page fence violations
- Attempts to alter the DMS registers while memory protect is enabled.

DMS errors only occur when MEM and memory protect are both enabled.

RTE-IV  
AND  
DYNAMIC MAPPING

## DYNAMIC MAPPING UNDER RTE

### USES:

====

- Allow many partitions in systems with >32K memory.
- Increase memory space for user programs by removing most of RTE, device drivers, SAM, memory resident programs, memory resident librarys and optionally common from the user's logical address space (or user map).
- Provide additional space for each program's base page links by using the base page fence.

### IMPLEMENTATION:

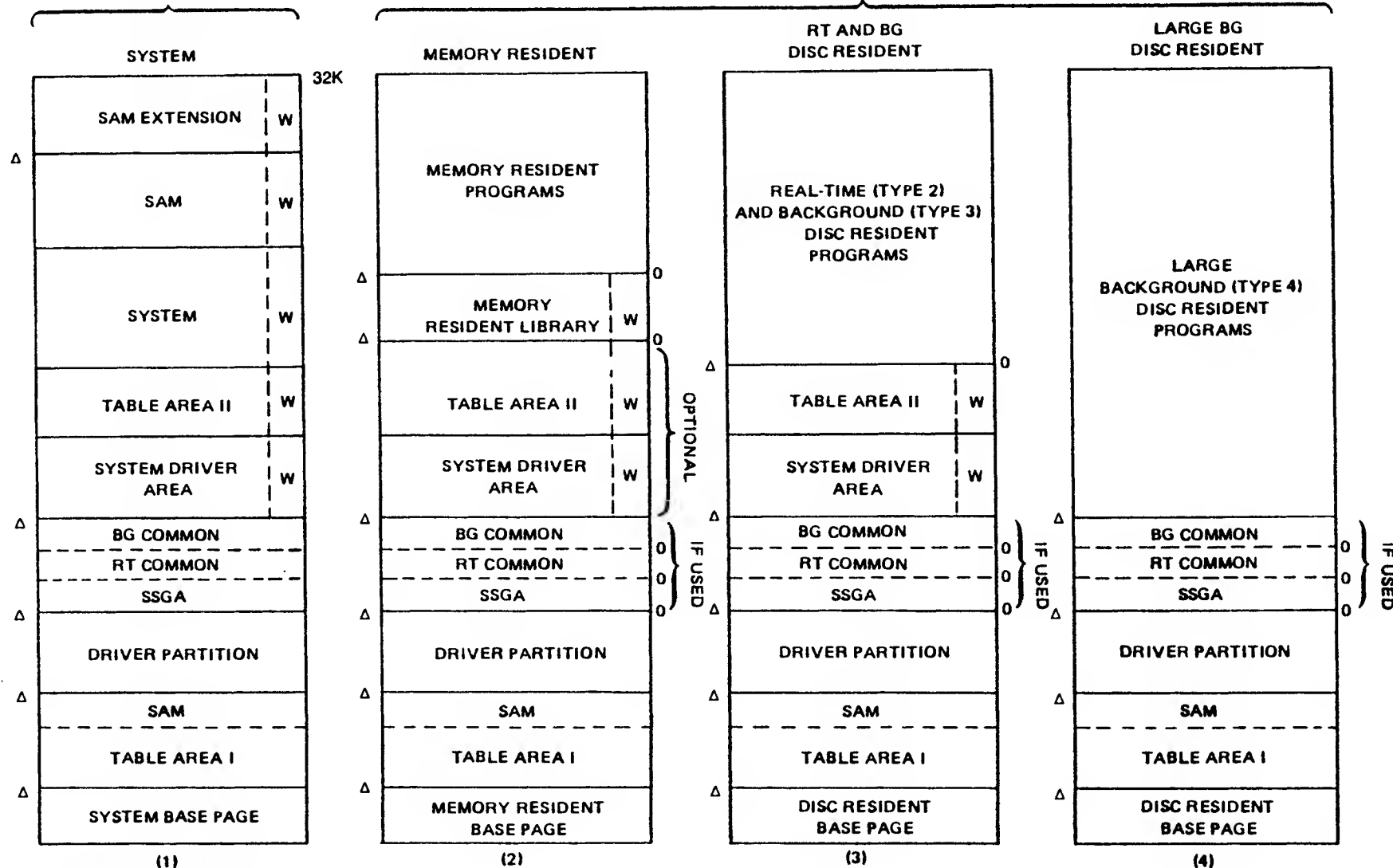
=====

- System map is fixed after boot up.
- The user map is built when a program is dispatched for the first time or dispatched for the first time after a swap-in. For a context switch, RTE saves and restores the user map in the unused portion of the partition base page. A copy of the memory resident map is kept in the system (\$MRMP).
- Separate map (a copy of the system or user map) for each DCPC channel.
- The base page fence register allows each user's logical base page to include part of the system base page.
- System map is automatically enabled upon all interrupts.
- DCPC maps are enabled/disabled on a word by word basis during DCPC processing.

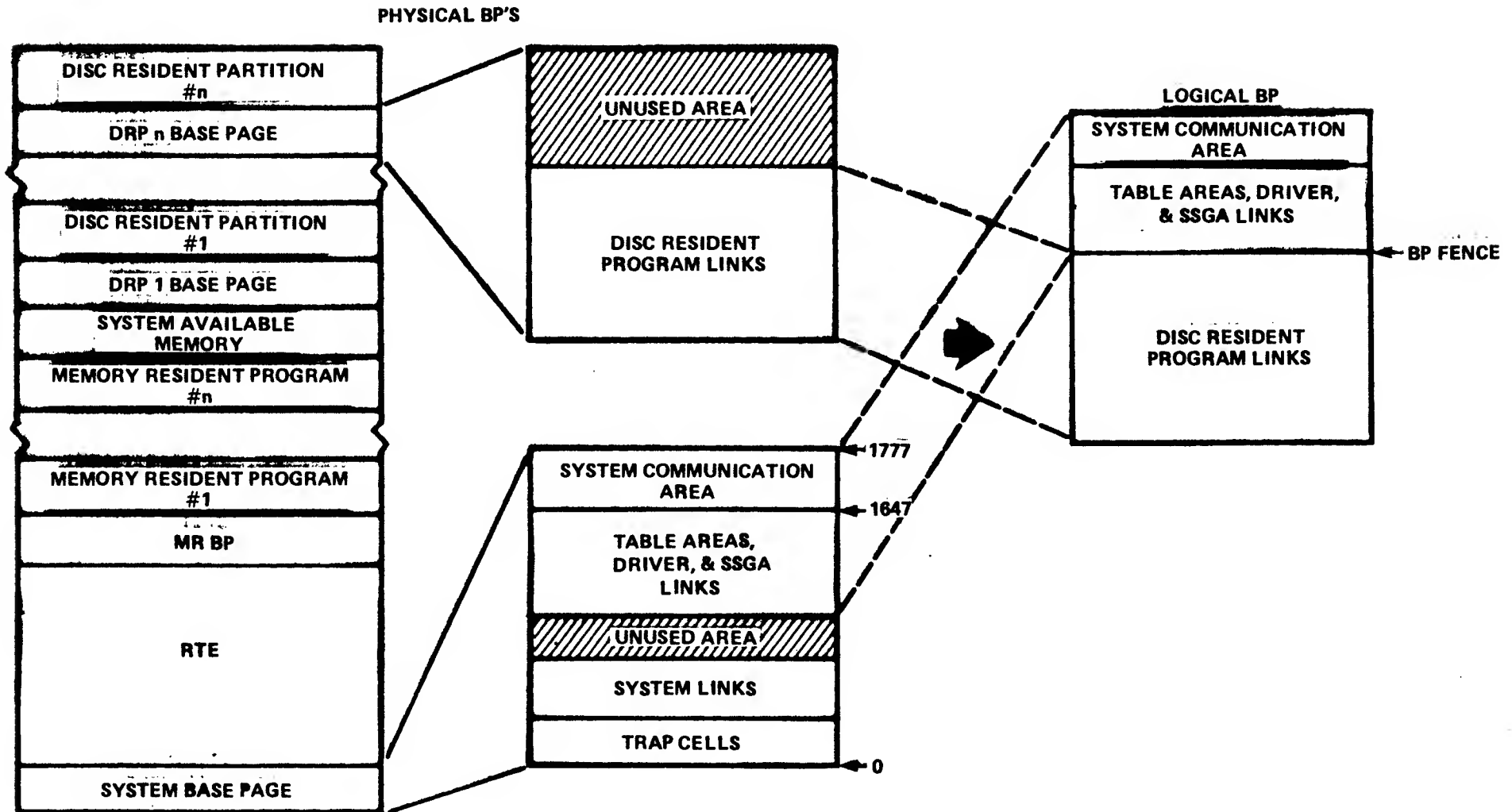
# RTE-IV LOGICAL MEMORY MAPS

DESCRIBED BY  
SYSTEM MAP

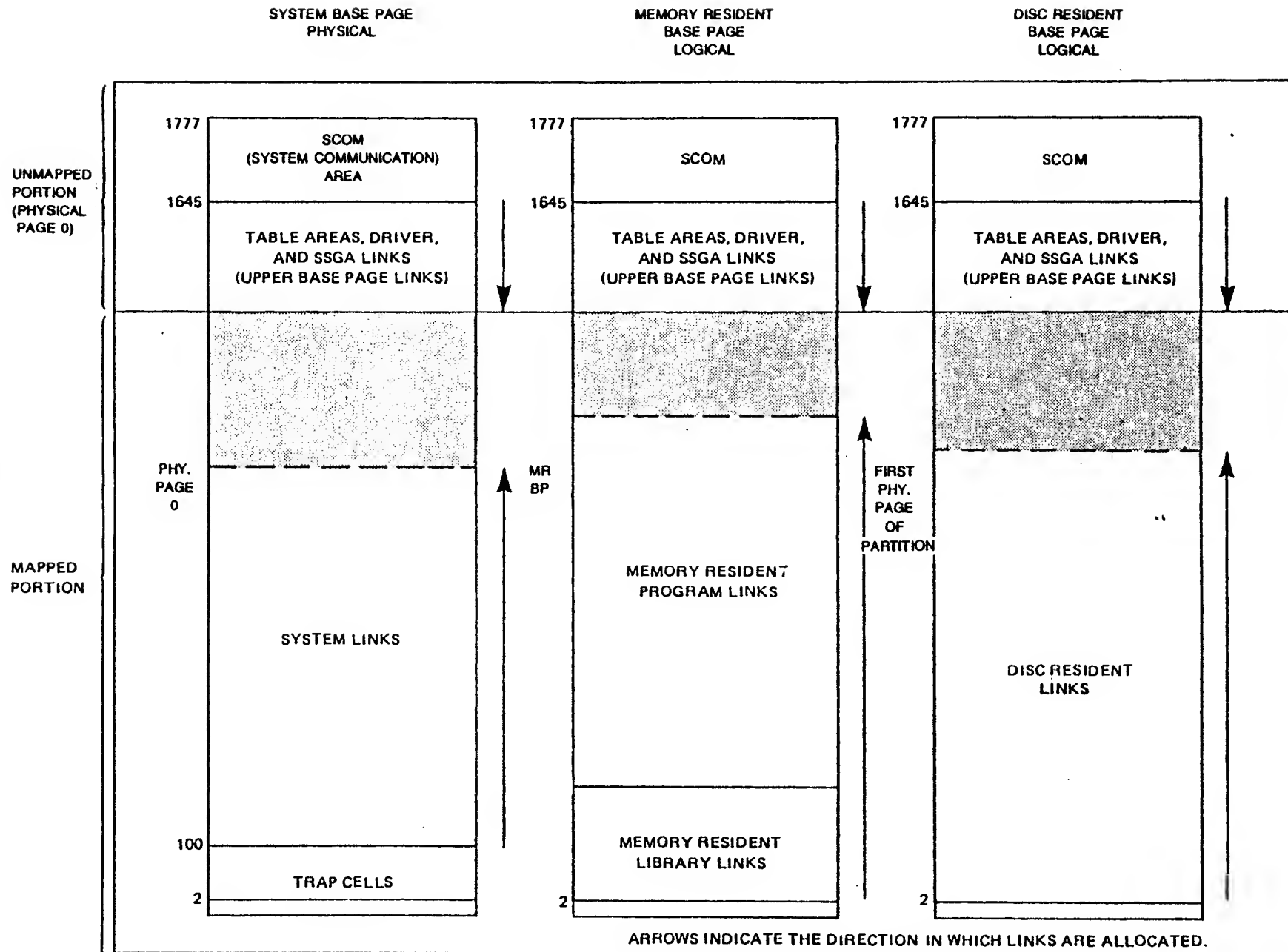
THREE POSSIBLE CONFIGURATIONS DESCRIBED  
BY USER MAP



# USER LOGICAL BASE PAGE



## BASE PAGES

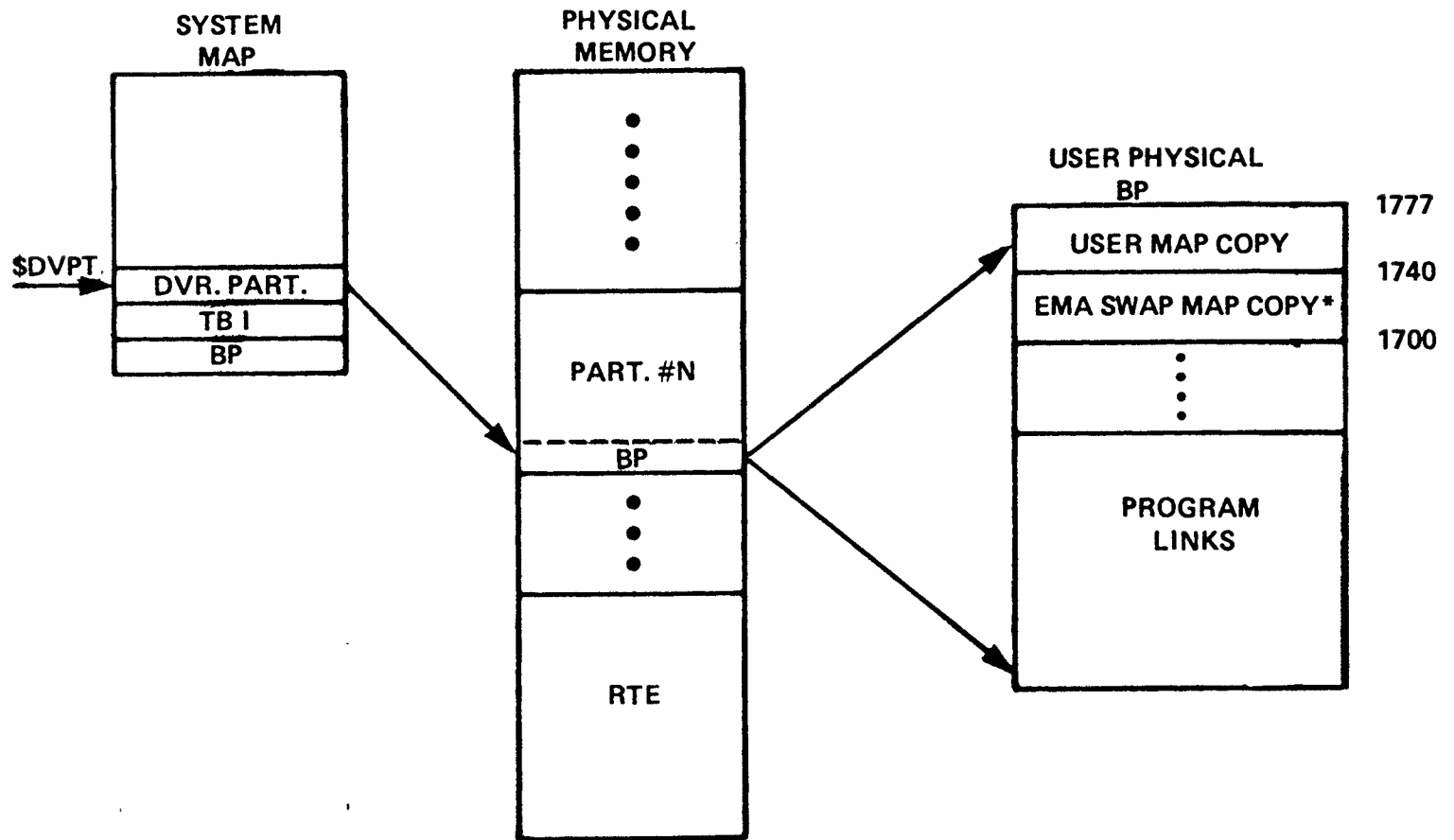


## MAP SETUP PROCEDURES

- Set M.P. fence from MPFT index in word 21 of I.D. seg.
- Set up appropriate user map: (\$SMAP)
  - 1) Load base page register from MAT entry, word #3 (start page.)
  - 2) Load system registers to map Table Area I and II, SDA, Driver Partition, and common based upon the program type (MPFI). The starting physical page is #1 for these areas and the ending page depends on the variables:  
  
\$CMST - starting logical page number of common (org. 0)  
\$SDA - starting logical page number of SDA (org. 0)  
\$SDT2 - # of pages in SDA and Table Area II
  - 3) At the next register map in the remainder of the partition by incrementing and loading until register number specified by ID word 21 is reached.
  - 4) Set remaining registers read/write protected.
- User map gets copied directly into DCPC maps when DCPC is used (swapping, DMA I/O)
- At boot-up \$STRT sets up the system map. \$ZZZZ initializes the remaining maps and sets the BP fence address.

# USER MAP SAVE AREA

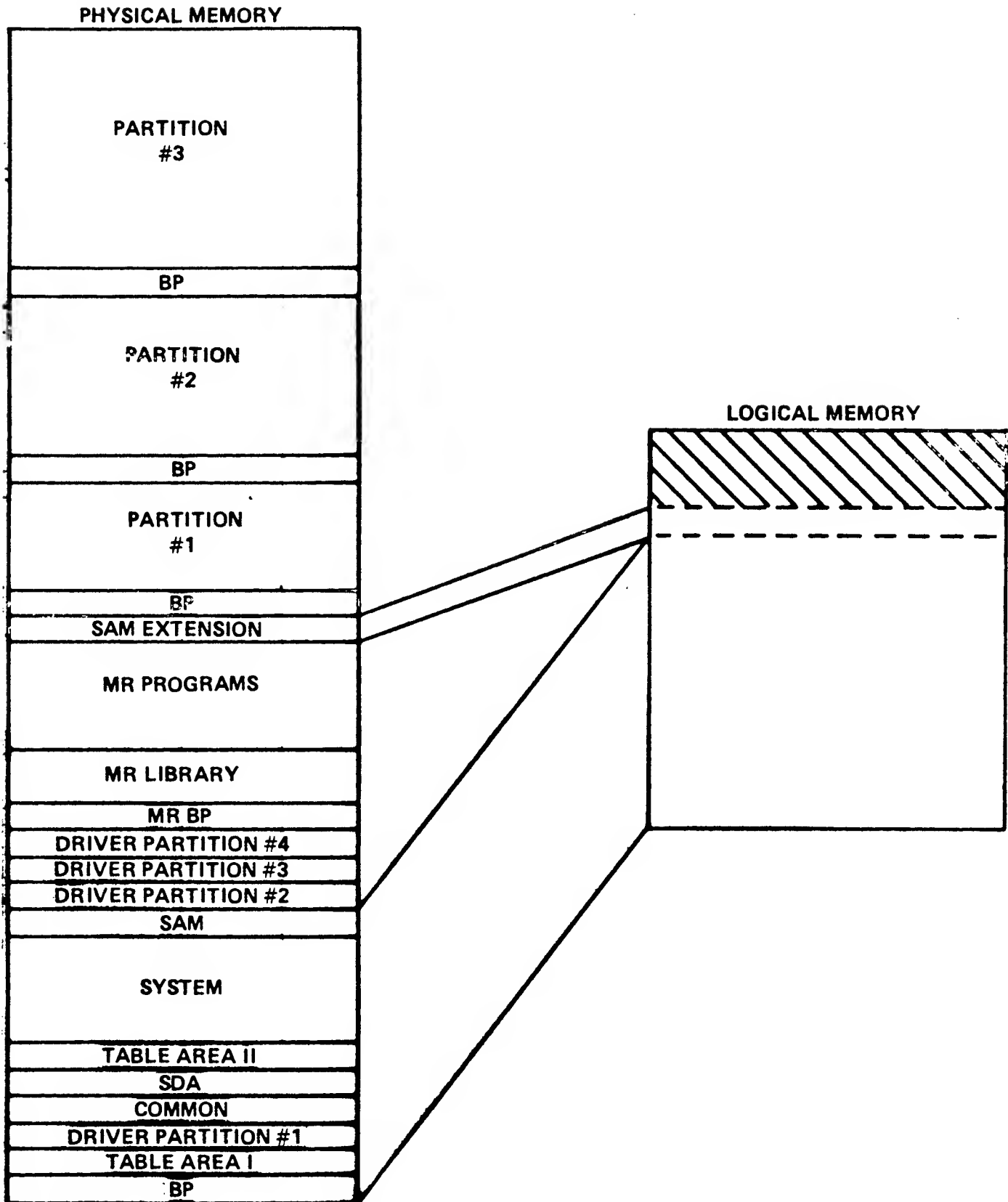
DISP4 & RTIO4 USE THE UPPER PORTION OF THE PHYSICAL BASE PAGE OF EACH PARTITION TO SAVE AND RESTORE EACH USER'S MAP. THE DRIVER PARTITION MAP REGISTER (\$DVPT) IS USED TO ACCESS THE USER'S PHYSICAL BP.



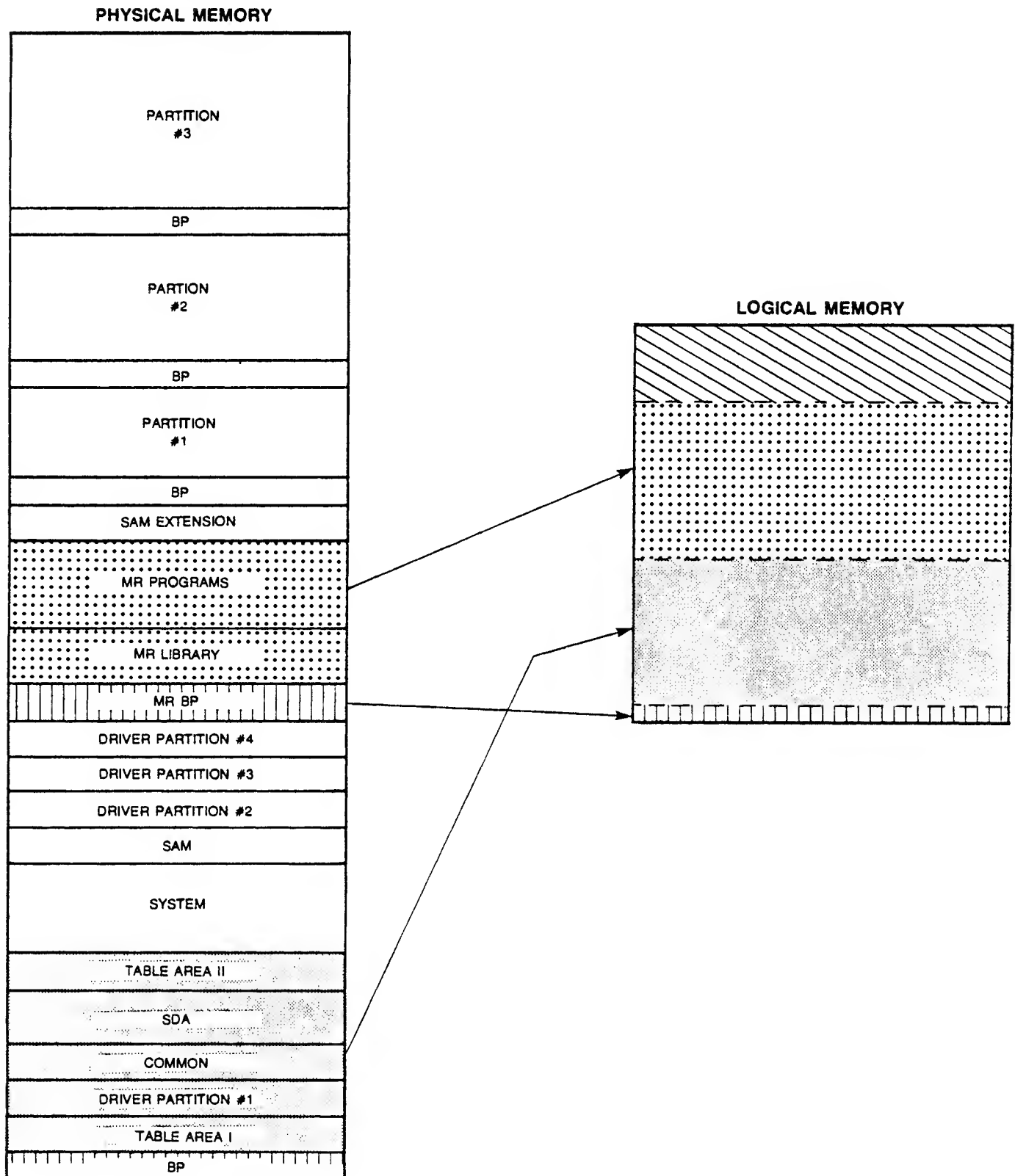
\* USED ONLY DURING THE SWAPPING OF THE EMA PORTION OF A PROGRAM. THIS PREVENTS THE DESTRUCTION OF THE ORIGINAL COPY OF THE USER MAP. AFTER THE PROGRAM HAS BEEN SWAPPED, THE USER MAP IS MODIFIED TO SWAP EMA CHUNKS EQUAL IN SIZE TO THE LOGICAL ADDRESS SPACE OF TYPE 4 PROGRAMS (27K MAX.).



# SAMPLE SYSTEM MAP



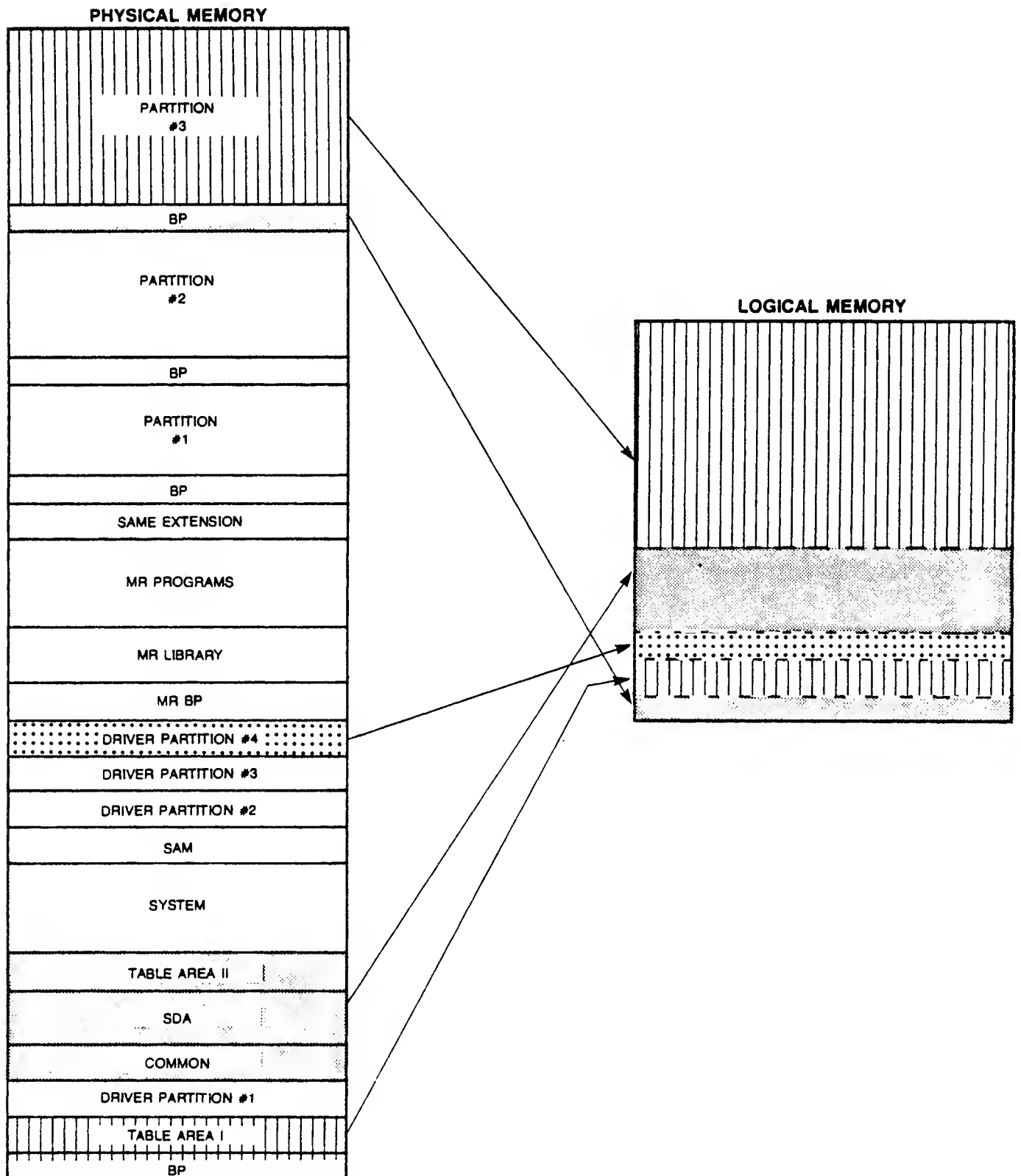
# SAMPLE USER MAP MEMORY RESIDENT PROGRAM WITH COMMON



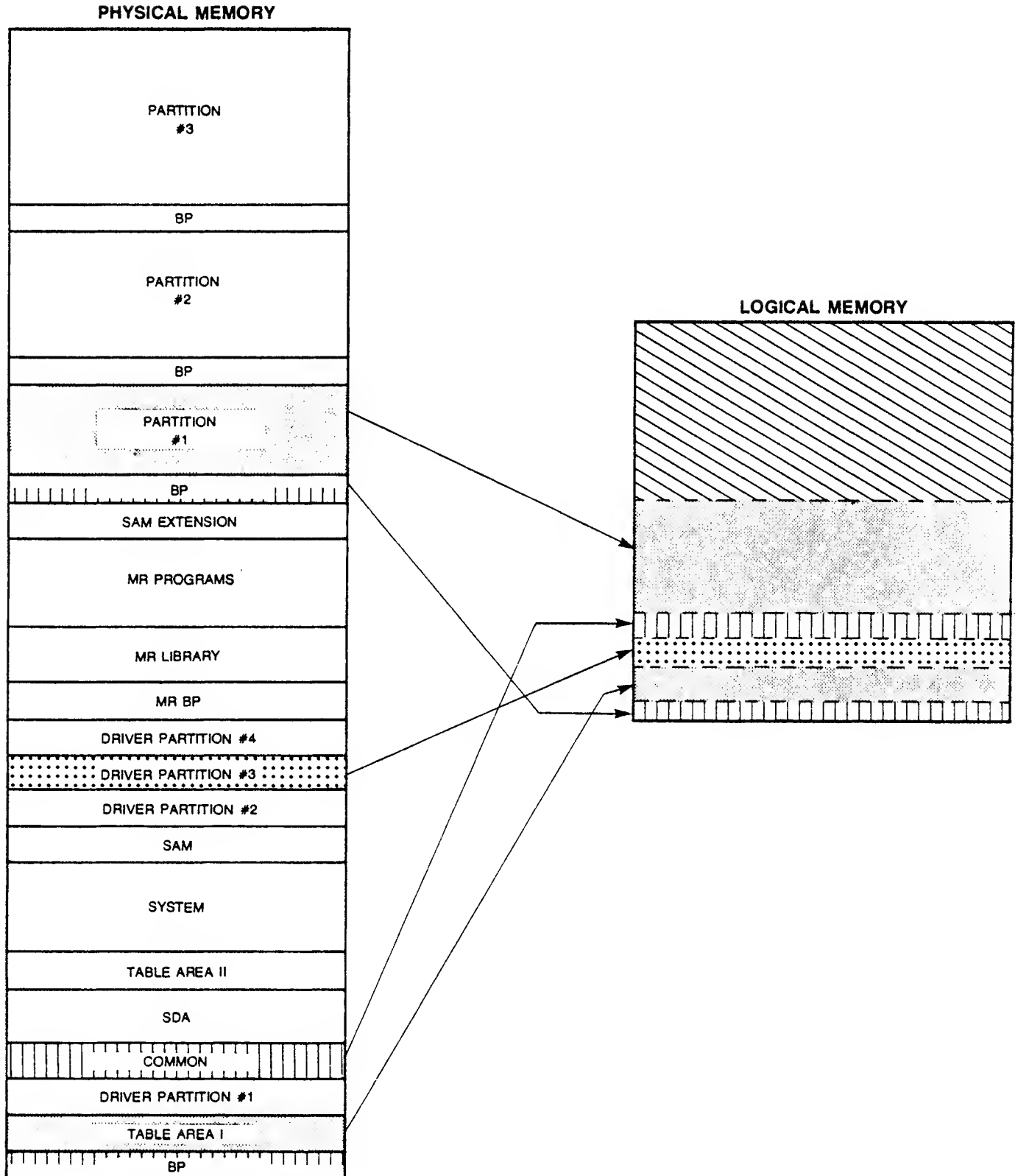
# SAMPLE USER MAP

## DISC RESIDENT PROGRAM WITH OR WITHOUT COMMON

### (TYPE 2 OR 3)



# SAMPLE USER MAP LARGE BG DISC RESIDENT PROGRAM WITH COMMON (TYPE 4)



# SAMPLE RTE-IV MAPS

## LIST OF DYNAMIC MAPPING REGISTERS FOR SYSTEMMAP

000000	000001	000057	000003	000004	040005	040006	040007
040010	040011	040012	040013	040014	040015	040016	040017
040020	040021	040022	040023	040024	040025	040026	040027
040054	040031	040032	040033	040034	040035	040036	040037

## LIST OF DYNAMIC MAPPING REGISTERS FOR USER MAP

000057	000001	000002	000003	000004	040005	040006	040007
040010	040011	040012	000060	000061	000062	140000	140001
140002	140003	140004	140005	140006	140007	140010	140011
140012	140013	140014	140015	140016	140017	140020	140021

## LIST OF DYNAMIC MAPPING REGISTERS FOR DCPC1 MAP

000057	000001	000002	000003	000004	040005	040006	040007
040010	040011	040012	000060	000061	000062	140000	140001
140002	140003	140004	140005	140006	140007	140010	140011
140012	140013	140014	140015	140016	140017	140020	140021

## LIST OF DYNAMIC MAPPING REGISTERS FOR DCPC2 MAP

000000	000001	000002	000003	000004	040005	040006	040007
040010	040011	040012	040013	040014	040015	040016	040017
040020	040021	040022	040023	040024	040025	040026	040027
040054	040031	040032	040033	040034	040035	040036	040037

STAT= 173446 VIOL= 000140

USER PROGRAM TYPE = 3  
\$CMST = 4  
\$SDA = 5  
\$STD2 = 6

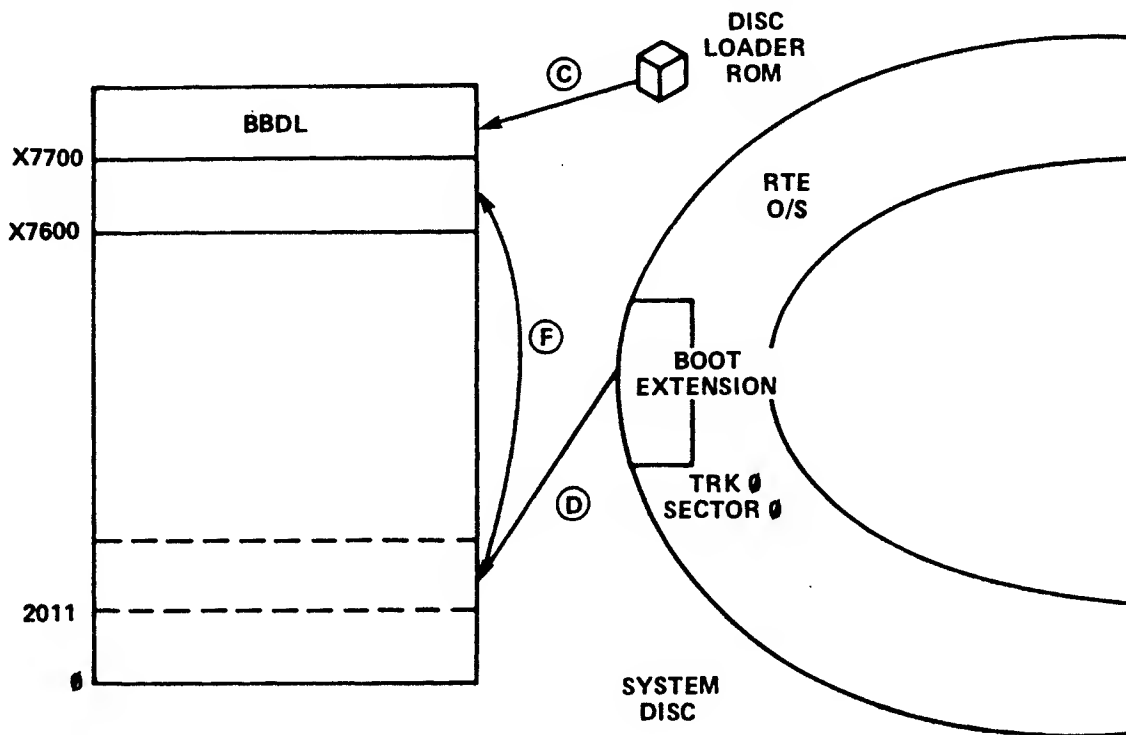


**RTE  
BOOT PROCESS**





# RTE BOOT PROCESS



- A. SET THE S REGISTER.
- B. PRESS PRESET TO DISABLE THE INTERRUPT SYSTEM.
- C. PRESS IBL TO CAUSE THE LOADER MICROPROGRAM TO READ THE DISC LOADER ROM (BBDL) INTO THE END OF UNMAPPED MEMORY.
- D. PRESS RUN AND BBDL WILL READ THE BOOT EXTENSION FROM DISC INTO MEMORY AND TRANSFER EXECUTION TO THE BOOT EXTENSION AT 2055 (OCTAL).
- E. BOOT EXTENSION OPTIONALLY HALTS WITH A 102077 IF S REGISTER BIT 5 IS SET REQUESTING RECONFIGURATION. ENTER SELECT CODES OF SYSTEM CONSOLE AND DISC. PRESS RUN TO CONTINUE.
- F. BOOT EXTENSION MOVES ITSELF TO END OF UNMAPPED MEMORY.

# 7905/7920 Loader ROM Program Listing

## 7905/20 DISC BOOT LOADER (12992B) - RPL COMPATIBLE

```

0001          ASMB,A,B,L
0003 07700          ORG 7700B
0004*****
0005*
0006*          REVISION          05 AUG 77*
0007*          PART NUMBER      12992-80002*
0008*          PRODUCT NUMBER   12992B*
0009*
0010*****
0011*
0012*  SWITCH REGISTER USAGE
0013*
0014*    15-14    LOADER SELECT
0015*    13       UNUSED
0016*    12       =0/1=RPL/MANUAL BOOT
0017*    11-6    DISC SELECT CODE
0018*    5-3     RESERVED
0019*    2-0     SUECHANNEL NUMBER
0020*
0021 00010          DC      EQU 10B
0022*
0023 07700 017727    START JSB STAT      GET STATUS
0024 07701 002021          SSA,RSS      IS DRIVE READY ?
0025 07702 027742          JMP DMA      YES, SET UP DMA
0026 07703 013714          AND B20      NO, CHECK STATUS BITS
0027 07704 002002          SZA          IS DRIVE FAULTY OR HARD DOWN ?
0028 07705 102030          HLT 30B      YES, HALT 30B, "RUN" TO TRY AGAIN
0029 07706 027700          JMP START    NO, TRY AGAIN FOR DISC READY
0030*
0031*  CONSTANTS
0032*
0033 07707 102011    ADDR1 OCT 102011 ← BOOT EXTENSION LOAD ADDRESS
0034 07710 102055    ADDR2 OCT 102055 ←
0035 07711 164000    CNT  DEC -6144
0036 07712 000007    D7   OCT 7        START ADDRESS
0037 07713 001400    STCMD OCT 1400
0038 07714 000020    B20  OCT 20
0039 07715 017400    STMSK OCT 17400
0040*  9 NOP'S
0044          LST
0045*
0046 07727 000000    STAT  NOP          STATUS CHECK SUBROUTINE
0047 07730 107710          CLC DC,C      SET STATUS COMMAND MODE
0048 07731 063713          LDA STCMD     GET STATUS COMMAND
0049 07732 102610          OTA DC        OUTPUT STATUS COMMAND
0050 07733 102310          SFS DC        WAIT FOR STATUS#1 WORD
0051 07734 027733          JMP *-1
0052 07735 107510          LIB DC,C      B-REG = STATUS#1 WORD
0053 07736 102310          SFS DC        WAIT FOR STATUS#2 WORD
0054 07737 027736          JMP *-1
0055 07740 103510          LIA DC,C      A-REG = STATUS#2 WORD

```

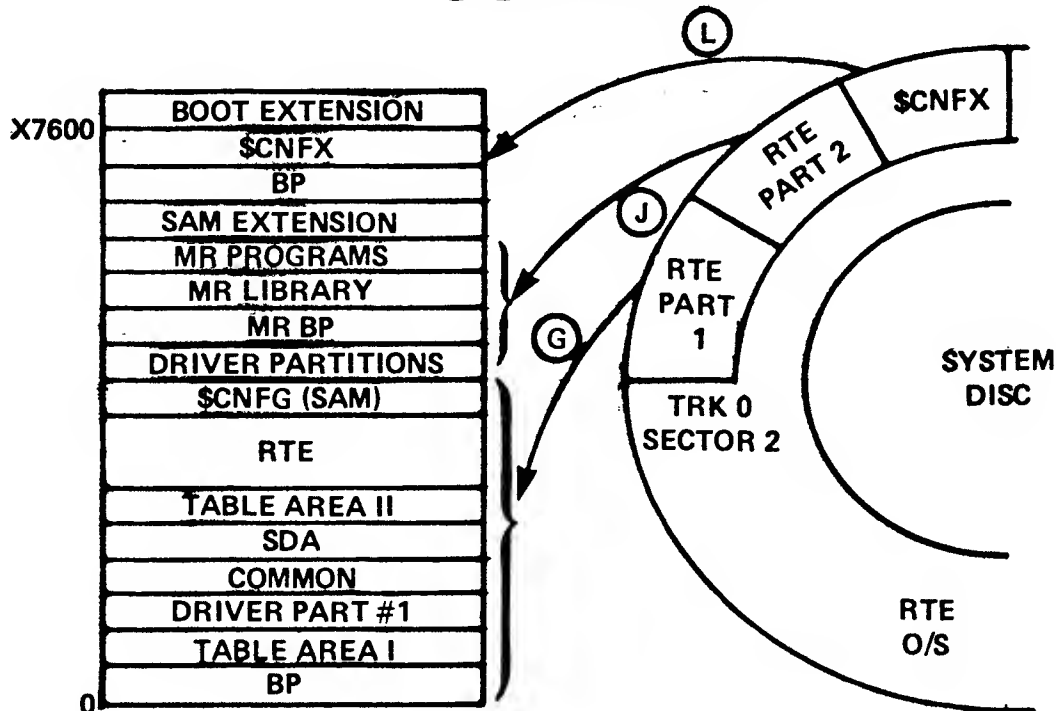
## 7905/7920 Loader ROM Program Listing (Continued)

```

0056 07741 127727      JMP STAT,I      RETURN
0057*
0058*  (SET UP DMA CHANNEL)
0059*
0060 07742 067776 DMA  LDB DMACW      GET DMA CONTROL WORD
0061 07743 106606      OTB 6          OUTPUT DMA CONTROL WORD
0062 07744 067707      LDB ADDR1      GET MEMORY ADDRESS
0063 07745 106702      CLC 2          SET MEMORY ADDRESS INPUT MODE
0064 07746 106602      OTB 2          OUTPUT MEMORY ADDRESS TO DMA
0065 07747 102702      STC 2          SET WORD COUNT INPUT MODE
0066 07750 067711      LDB CNT        GET WORD COUNT ← ONE TRACK
0067 07751 106602      OTB 2          OUTPUT WORD COUNT TO DMA
0068*FALL THRU
0069* 7905/20 COLD LOAD COMMAND
0070*
0071 07752 106710 CLDLE CLC DC      SET COMMAND INPUT MODE
0072 07753 102501      LIA 1          LOAD SWITCH
0073 07754 106501      LIB 1          REGISTER SETTINGS
0074 07755 013712      AND D7        ISOLATE HEAD NUMBER
0075 07756 005750      BLF,CLE,SLB   BIT 12=0?
0076 07757 027762      JMP *+3       NO,MANUAL BOOT
0077 07760 002002      SZA          YES,RPL BOOT. HEAD#=0?
0078 07761 001000      ALS          NO,HEAD#=1, MAKE HEAD#=2
0079 07762 001720      ALF,ALS       FORM COLD LOAD
0080 07763 001000      ALS          COMMAND WORD
0081 07764 103706      STC 6,C       (ACTIVATE DMA)
0082 07765 103610      OTA DC,C      OUTPUT COLD LOAD COMMAND
0083 07766 102310      SFS DC        IS COLD LOAD COMPLETED ?
0084 07767 027766      JMP *-1       NO, WAIT
0085 07770 017727      JSB STAT      YES, GET STATUS
0086 07771 060001      LDA 1
0087 07772 013715      AND STMSK     A-REG = STATUS BITS OF STATUS#1 WORD
0088 07773 002002      SZA          IS TRANSFER OK ?
0089 07774 027700      JMP START     NO,TRY AGAIN
0090 07775 117710 EXIT JSB ADDR2,I  YES, (EXECUTE LOADED PROGRAM @ 2055B)
0091*FALL THRU
0092* THE NEXT 2 WORDS MUST BE THE LAST 2 WORDS
0093* IN THE BOOTSTRAP LOADER IN THE LAST 2 MEMORY LOCATIONS
0094 07776 000010 DMACW ABS DC
0095 07777 170100      ABS -START
0096                                END

```

# BOOT PROCESS CONTINUED



- G. BOOT EXTENSION LOADS RTE FROM BASE PAGE THRU \$CNFG.
- H. CONTROL IS TRANSFERRED TO RTE AT \$STRT WITH A JMP 3,I
- I. \$STRT SETS UP AND ENABLES THE SYSTEM MAP
- J. \$CNFG LOADS DRIVER PARTITIONS, MEMORY RESIDENT BP, MR LIBRARY, AND MR PROGRAMS INTO MEMORY.
- K. \$CNFG MAKES SELECT CODE RE-ASSIGNMENTS, IN TRAP CELLS, INTERRUPT TABLE, AND EQTS.
- L. \$CNFG SETS UP USER MAP FOR \$CNFX AS A TYPE 3 PROGRAM AND LOADS \$CNFX.

# 7905 BOOT EXTENSION

0001	0002*				
0002	0003*				
0003	0004*				
0004	0006	07600	063725	START LDA HIGH	HIGH CORE ADDRESS(INIT. AT 2011B)
0005	0007	07601	003300	CMA,CCE	SET DIRECTION BIT
0006	0008	07602	073741	STA RECNT	INIT COUNT
0007	0009	07603	005500	ERB	100000B IS LOW CORE ADDRESS
0008	0010	07604	106702	CLC 2	WITH DIRECTION BIT SET
0009	0011	07605	106602	OTB 2	SET MEMORY ADDRESS REGISTER
0010	0012	07606	063733	LDA SC	
0011	0013	07607	002003	SZA,RSS	COMING FROM PAPER TAPE BOOT?
0012	0014	07610	102501	LIA 1	YES,GET CONTENTS OF S REGISTER
0013	0015	07611	101046	LSR 6	
0014	0016	07612	013753	AND B77	MASK SELECT CODE OF DISC
0015	0017	07613	073733	STA SC	SAVE IT
0016	0018	07614	163731	LOOP LDA HDA,I	CONFIGURE I/O INSTR FROM STIO
0017	0019	07615	167731	LDB HDA,I	
0018	0020	07616	013751	AND IOMSK	MASK OUT LOWER 6 BITS IN INSTR
0019	0021	07617	047733	ADB SC	CONFIGURE INSTR FOR DISC SC
0020	0022	07620	053746	CPA IOG	IS THIS INSTR IN I/O GROUP?
0021	0023	07621	177731	STB HDA,I	YES, THEN STORE IT BACK
0022	0024	07622	037731	ISZ HDA	MOVE ON TO NEXT INSTR
0023	0025	07623	063731	LDA HDA	
0024	0026	07624	053734	CPA HDA3	ALL DISC I/O INSTR CONFIGURED?
0025	0027	07625	002405	CLA,INA,RSS	YES, SET A TO 1 FOR SECTOR #
0026	0028	07626	027614	JMP LOOP	NO THEN CONFIGURE NEXT ONE
0027	0029	07627	073755	SLOAD STA BENT	
0028	0030	07630	063754	LDA T#ACK	
0029	0031	07631	006400	CLB	
0030	0032	07632	100400	DIV #HDS	GET RELATIVE TRACK/HEAD
0031		07633	007747		
0032	0033	07633		DDIV EQU *-1	
0033	0034	07634	043743	ADA TBASE	ADD TRACK ZERO TO GET ABS. TRACK
0034	0035	07635	073730	STA CYLA1	SAVE FOR ADDRESSING
0035	0036	07636	073733	STA CYLA3	SAVE FOR ADDRESSING
0036	0037	07637	047750	ADB BHD4	ADD THE BASF HEAD ADDRESS
0037	0038	07640	063755	LDA BENT	GET SECTOR
0038	0039	07641	005727	BLF,BLF	PUT HEAD IN HIGH 8 AND
0039	0040	07642	047755	ADB BENT	ADD THE SECTOR
0040	0041	07643	002001	RSS	SKIP OVER ADDRESS OF BENT
0041	0042	07644	002166	OCT 2166	DEFINE ADDR. OF BENT(INIT. AT 205
0042	0043	07645	077731	STB HDA	SET THE HEAD/SECTOR ADDRESSES
0043	0044	07646	077734	STB HDA3	
0044	0045	07647	100047	LSL 7	SECTOR TIMES 128
0045	0046	07650	003004	CMA,INA	AND SUBTRACT FORM
0046	0047	07651	043740	ADA #WDTK	NUMBER OF WORDS PER TRACK
0047	0048	07652	073724	STA P#WDS	SET POSITIVE # WORDS
0048	0049	07653	003004	CMA,INA	AND
0049	0050	07654	073725	STA N#WDS	NEG. # WORDS THIS TRACK
0050	0051	07655	063741	LDA RECNT	GET # LEFT
0051	0052	07656	002021	SSA,RSS	IF POSITIVE
0052	0053	07657	124003	JMP 3,I	DONE, SO EXIT
0053	0054*				
0054	0055	07660	043724	ADA P#WDS	ELSE SET TO READ
0055	0056	07661	073741	STA RECNT	SAVE REMAINING COUNT
0056	0057	07662	002020	SSA	NEXT TRACK
0057	0058	07663	002400	CLA	USE MIN. OF # ON TRACK OR
0058	0059	07664	043725	ADA N#WDS	NUMBER LEFT

0059	0060	07665	102702	STC 2	SET DMA FOR WORD COUNT
0060	0061	07666	102602	UTA 2	AND SENT IT
0061	0062	07667	067742	LDB D#PRM	GET THE COMMAND
0062	0063	07670	160001	SLOOP LDA 1,I	
0063	0064	07671	001275	RAL,CLE,SLA,ERA	IF SIGN BIT SET
0064	0065	07672	106700	DSK10 CLC 0	SEND COMMAND IS COMMING
0065	0066	07673	103600	DSK11 DTA 0,C	SEND THE COMMAND
0066	0067	07674	057744	CPB A#DMA	IF DMA
0067	0068	07675	103706	STC 6,C	START IT
0068	0069	07676	102700	DSK12 STC 0	ALLOW ATTENTION
0069	0070	07677	006045	SEZ,INB,RSS	IF NOT A COMMAND
0070	0071	07700	027703	JMP STDMA	DON'T WAIT FOR FLAG
0071	0072*				
0072	0073	07701	102300	DSK13 SFS 0	WAIT FOR THE FLAG
0073	0074	07702	027701	JMP *-1	
0074	0075	07703	102106	STDMA STF 6	STOP DMA IF NEEDED
0075	0076	07704	057745	CPB A#END	END OF LOOP?
0076	0077	07705	002001	RSS	SKIP IF END
0077	0078	07706	027670	JMP SLOOP	NOT END AROUND WE GO
0078	0079*				
0079	0080	07707	103500	DSK14 LIA 0,C	GET STATUS 1
0080	0081	07710	102300	DAK15 SFS 0	WAIT FOR FLAG
0081	0082	07711	027710	JMP *-1	
0082	0083	07712	107500	DKS16 LIB 0,C	GET STATUS 2
0083	0084	07713	013723	AND C174B	ISOLATE
0084	0085	07714	002003	SZA,RSS	IF NO ERRORS
0085	0086	07715	027721	JMP OK	CONTINUE
0086	0087*				
0087	0088	07716	101100	SWP	SWITCH A AND B REG. CONTENTS
0088	0089	07717	102031	HLT31 HLT 31B	ELSE HALT
0089	0090	07720	027717	JMP HLT31	TRY AGAIN
0090	0091*				
0091	0092	07721	037754	OK ISZ T#ACK	STEP THE TRACK ADDRESS
0092	0093	07722	027627	JMP SLOAD	GO LOAD(A=0=SECTOR ADDRESS)
0093	0094*				
0094	0095	07723	017400	C174B OCT 17400	
0095	0096	07724	177600	P#WDS DEC -128	
0096	0097	07725	077477	N#WDS OCT 77477	
0097	0098	07725		HIGH EQU N#WDS	
0098	0099	07726	113000	WAK OCT 113000	
0099	0100	07727	101200	SKCMD OCT 101200	
0100	0101	07730	077600	CYLA1 OCT 77600	
0101	0102	07731	077672	HDA OCT 77672	
0102	0103	07732	106000	AD#RC OCT 106000	
0103	0104	07733	000000	CYLA3 NOP	
0104	0105	07733		SC EQU CYLA3	
0105	0106	07734	077713	HDA3 OCT 77713	
0106	0107	07735	107404	FILM# OCT 107404	
0107	0108	07736	102400	R#CMD OCT 102400	
0108	0109	07737	101400	S#TAC OCT 101400	
0109	0110	07740	014000	#WDTK DEC 6144	
0110	0111	07741	077600	RECNT OCT 77600	CONFIGURED TO BBL ADDRESS
0111	0112	07742	077726	D#PRM OCT 77726	
0112	0113	07743	000000	TRASE NOP	FIRST TRACK#-MUST BE AT START+143
0113	0114	07744	077736	A#DMA OCT 77736	
0114	0115	07745	077740	A#END OCT 77740	
0115	0116	07746	102000	IOG OCT 102000	
0116	0117	07747	000002	#HDS DEC 2	# SURFACES
0117	0118	07750	000000	BHD# NOP	STARTING HEAD #
0118	0119	07751	172076	IOMSK OCT 172076	

0119	0120	07752 002011	SPCAD	OCT 2011	
0120	0121	07753 000077	B77	OCT 77	
0121	0122	07754 000000	T#ACK	NOP	
0122	0123*				
0123	0124*				
0124	0125*				
0125	0126*				
0126	0127	07755 000000	BENT	NOP	JSB HERE FROM BBDL(INIT. AT 2166B
0127	0128	07756 102106		STF 6	CLEAN UP DMA
0128	0129	07757 107700		CLC 0,C	AND THE I/O SYSTEM
0129	0130	07760 006400		CLB	ELIMINATE HLT 77B
0130	0131	07761 102501		LIA 1	READ S REG.
0131	0132	07762 072144		STA SC	SAVE S REG. CONTENTS
0132	0133	07763 101045		LSR 5	
0133	0134	07764 002011		SLA,RSS	WAS BIT 5 OF S REG. SET?
0134	0135	07765 026201		JMP NORCN	NO, THEN RECONFIG. NOT REQD
0135	0136	07766 102077		HLT 77B	YES, THEN HALT TO LET USER SET S
0136	0137	07767 026202		JMP DRBOT	RELOCATE THE REST OF THIS BOOT
0137	0138	07770 106601	NORCN	OTB 1	CLEAR S REG.
0138	0139*				
0139	0140	07771 162163	DRBOT	LDA SPCAD,I	MOVE 128 WORDS TO BBL-128
0140	0141	07772 172152		STA RECNT,I	
0141	0142	07773 036163		ISZ SPCAD	
0142	0143	07774 036152		ISZ RECNT	
0143	0144	07775 036135		ISZ P#WDS	DONE?
0144	0145	07776 026202		JMP DRBOT	NO, GET NEXT WORD
0145	0146*				
0146	0147	07777 126141		JMP CYLA1,I	YES,GO EXECUTE THE BOOT
0147	0148*				
0148	0150			END START	

BOOT PROCESS  
COMPLETION  
(SYSTEM START UP)

- A. \$CNFX - Redefine and remap SAM extension including up to 5 bad pages. Accept changes to partitions, program sizes, and partition assignments. Record in memory and optionally on disc.
- B. \$STRT - Set up SAM by calling \$RTN. EQT 1 thru EQT 6 on BP contain SAM block addresses and sizes. EQT1&2 are SAM default block. EQT3&4 are SAM extension and EQT5&6 are SAM in Table Area I.
- C. \$ZZZZ - Disable interrupt system
- D. \$ZZZZ - Subroutine MPINT sets up remaining three maps, base page fence, and MAT table linked lists.
- E. \$ZZZZ - Schedule FMGR to set up the file system.
- F. \$STRT - Save ID segment addresses of FMGR, D.RTR, EDIT, and SMP.
- G. \$SCLK - Start the TBG and print "SET TIME".
- H. \$XEQ - Idle loop.



LOADED AT 2011B

BOOT START ADDRESS (2055B)

LWA OF RTE FROM BP THRU \$CNFG

BOOT EXTENSION  
RELOCATION ADDRESS

START  
(2166B)

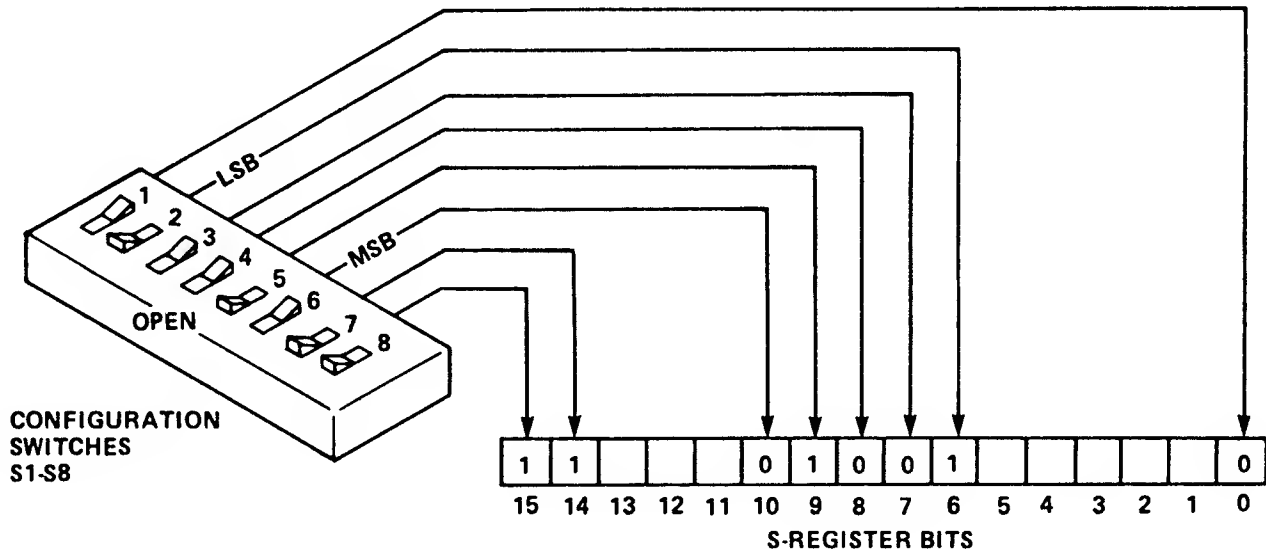
RECONFIGURATOR HALT  
(BIT 5 SET)[illegible]

5-9

# REMOTE PROGRAM LOADING RPL

RTE IS AUTOMATICALLY BOOted WHEN POWER IS TURNED ON AND THE LOCK/OPERATE SWITCH IS IN THE LOCK POSITION.

S-REGISTER BOOT INFORMATION IS OBTAINED FROM THE CONFIGURATION SWITCHES MOUNTED ON THE CPU BOARD.



SWITCH	FUNCTION
S1	<p>7905/7920 DISC  OPEN POSITION SELECTS HEAD 000, REMOVABLE PLATTER  CLOSED POSITION SELECTS HEAD 010, FIXED PLATTER  ON 7905</p> <p>9885 FLEXIBLE DISC  MUST BE IN CLOSED POSITION. ALL LOADING STARTS AT  TRACK 0, SECTOR 1</p> <p>7900/7901 DISC  OPEN POSITION SELECTS FIXED PLATTER  CLOSED POSITION SELECTS REMOVABLE PLATTER</p>
S2 S3 S4 S5 S6	<p>DECODES THE FIVE BIT OCTAL SELECT CODE OF THE INPUT  DEVICE (S.C. 10-37) SWITCHES SHOWN SET TO SELECT CODE  11<sub>8</sub> (01001<sub>2</sub>)</p>
S7	<p>IF RPL ENABLED, OPEN POSITION SELECTS ROM 10; CLOSED  POSITION SELECTS ROM 11</p>
S8	<p>OPEN POSITION RPL NOT ENABLED  CLOSED POSITION RPL ENABLED</p>

**DBUGR/CMM4**



## DBUGR FEATURES

Used to debug programs under RTE-IV

- \* Symbolic or octal printout
- \* Register examination and change
- \* Memory search
- \* Memory clear
- \* Breakpoint
- \* Map examination.

## DBUGR EXECUTION

- \* Relocating with program to be debugged

IN-LINE:     FTN4,L,M

          PROGRAM PROGA

          CALL DBUGR [or CALL DBUGR(LU)]

          .

          .

          .

          Where LU = lu # of console if omitted,  
          LU is defaulted to terminal from which  
          the program was scheduled.

          OR

WITH LOADR:     \*RU,LOADR,,%PROGA,,DB

- \* Starting DBUGR

          \*RU,PROGA

          START DBUGR ← (DBUGR PRINTOUT)

          (DBUGR IS READY TO ACCEPT COMMANDS)

# DBUGR SAMPLE PROGRAM (SOURCE)

This program will be used for the DBUGR command examples

```
0001  FTH4,L,M
0002      PROGRAM PROGA
0003      DIMENSION IARY(10)
0004      CALL DBUGR
0005      5  I = 10
0006      J = 3
0007      JJ = 2HAB
0008      X = 2.3
0009      DO 10 I = 1,10
0010      IARY(I) = I
0011      10 CONTINUE
0012      GO TO 5
0013      END
```

# DBUGR

## SAMPLE PROGRAM

### (MIXED LISTING)

```

0002      PROGRAM PROGA
0003      DIMENSION IARY(10)
0004      CALL DBUGR

                                IARY  BSS 00012B
                                00012 *000000      NOP
                                00013 000001X      JSB CLR10
                                00014 000015R      DEF *-2+00003B
0005      5 I = 10
                                00015 000002X      JSB DBUGR
                                00016 000017R      DEF *-4+00005B
0006      J = 3
                                00017 000050R @5      LDA 00050B
                                00020 000051R      STA I
0007      JJ = 2HAB
                                00021 000053R      LDA 00053B
                                00022 000052R      STA J
0008      X = 2.3
                                00023 000055R      LDA 00055B
                                00024 000054R      STA JJ
0009      DO 10 I = 1, 10
                                00025 000003X      JSB .DLD
                                00026 000060R      DEF 00060B
                                00027 000004X      JSB .DST
                                00030 000056R      DEF X
                                00031 000062R      LDA 00062B
                                00032 000051R      STA I
0010      IARY(I) = I
                                00033 000051R      LDA I
                                00034 000063R      ADA 00063B
0011      10 CONTINUE
                                00035 000064R      STA A.001
                                00036 000051R      LDA I
                                00037 100064R      STA A.001, I
0012      GO TO 5
                                00040 000051R @10      LDA I
                                00041 000062R      ADA 00062B
                                00042 000051R      STA I
                                00043 003004      CMA, INA
                                00044 000050R      ADA 00050B
                                00045 002021      SSA, RSS
                                00046 000033R      JMP 00033B
0013      END
                                00047 000017R      JMP @5
                                00050 000012      OCT 000012
                                I      BSS 00002B
                                00053 000003      OCT 000003
                                JJ     BSS 00001B
                                00055 040502      OCT 040502
                                X      BSS 00002B
                                00060 044631      OCT 044631
                                00061 115004      OCT 115004
                                00062 000001      OCT 000001
                                00063 177777R      DEF 77777B
                                A.001  BSS 00001B

```

\*NOTE: These offsets maybe used along with the program's load address to determine the final relocated address of each instruction. For example the final address of the "STA A.001" instruction would be 26002 (see PROGA load map) +35 or 26037.



# DBUGR

## SAMPLE PROGRAM

### (SYMBOL TABLE)

### FROM COMPILER

#### SYMBOL TABLE

NAME	ADDRESS	USAGE	TYPE	LOCATION
@10	00040R	STATEMENT NUMB		
@5	00017R	STATEMENT NUMB		
CLR10	00001X	STATEMENT FUNCTION	REAL	EXTERNAL
DBUGR	00002X	STATEMENT FUNCTION	REAL	EXTERNAL
I	00051R	VARIABLE	INTEGER	LOCAL
IARY	00000R	ARRAY(*)	INTEGER	LOCAL
J	00052R	VARIABLE	INTEGER	LOCAL
JJ	00054R	VARIABLE	INTEGER	LOCAL
X	00056R	VARIABLE	REAL	LOCAL

#### (LOAD MAP)

PROGA 26002 26056

CLR10	26067 26070	750701 24998-16001
DBUGR	26075 33734	92067-16075 REV.1805 780214
LOGLI	33735 34004	92067-16035 REV.1805 771117
IFPRK	34005 34034	92067-16035 REV.1805 770621

5 PAGES RELOCATED      5 PAGES PFD'D      NO PAGES EMA  
/LOADR:PROGA READY

/LOADR:SEND

**NOTE:** The address of each variable maybe found by adding PROGA's load address (26002) to the variables offset from the symbol table.

For example:

J	26002+52	=	26054
IARY	26002+0	=	26002

# CONVENTIONS FOLLOWED IN THE DBUGR EXAMPLES

—	User inputs are underlined.
CR	Carriage return
LF	Line feed ("Control J" on 2645/2648)
[ ]	Input control character which is not echoed
\	Escape (or ALT MODE on some terminals)
377	Octal number
377.	Decimal number

## DBUGR COMMANDS (BREAK POINTS)

n\B	Set the breakpoint at location n and clear the previous breakpoint. When the breakpoint location is encountered, control is transferred to DBUGR <u>prior</u> to execution of the instruction. Only one breakpoint is allowed at a time.
\P	Proceed with program execution until the next breakpoint
n\P	Proceed until the breakpoint is executed n octal times.
\T	Execute or trace the current instruction and break (single-step)
n\T	Trace (single-step) the next n octal instructions and break.
n\G	Continue execution at location n
\B	Remove the breakpoint

where,

\ = ESCAPE

# DBUGR EXAMPLES

## (BREAK POINTS)

\*RU,PROGA

START DBUGR

26002+34\B                    - set breakpoint in "do loop"  
\P                                - proceed  
 26036(ADA 26065) 1      115004      160010      114011      0 [CR]

(break-)(instruction) ( A ) ( B ) ( X ) ( Y ) ( EO )  
 (point address) ( reg. ) ( reg. ) ( reg. ) ( reg. ) ( reg. )

printout when breakpoint encountered

3\P                                - break after 3 executions of the breakpoint instruction  
 26036(ADA 26065) 4      115004      160010      114011      2 [CR]  
                                      breakpoint encountered again

\I                                - execute one instruction  
 26037(STA 26066) 26005      115004      160010      114011      2 [CR]

3\I                                - execute three instructions  
 26040(LDA 26053) 26005      115004      160010      114011      2  
 26041(STA 26066,I) 4      115004      160010      114011      2  
 26042(LDA 26053) 4      115004      160010      114011      2 [CR]

26002+20\B                    - reset breakpoint  
26002+17/G                    - begin execution at 26021  
 26022(STA 26053) 12      115004      160010      114011      2 [CR]

\B                                - remove breakpoint

\P                                - proceed

END DBUGR                    - program continues execution no longer under DBUGR control

# **DBUGR COMMANDS**

## **(MEMORY MODIFICATION)**

<code>n &lt; s:</code>	Define the symbol <code>s</code> as the value <code>n</code>
<code>n/</code>	Print and open for modification location <code>n</code>
<code>[CR]</code>	Close the open location
<code>n [CR]</code>	Store <code>n</code> (assembly instruction or octal constant) in the open location
<code>[LF]</code>	Print and open the next location
<code>n [LF]</code>	Same as <code>n [CR]</code> and also print and open the next location.
<code>n "[CR] or n" [LF]</code>	Same as above except <code>n</code> is interpreted as one or two ASCII characters.

# EXAMPLES

## (MEMORY MODIFICATION)

:RU,PROGA

START DBUGR

26002<R1:[CR]

- set "R1" equal to PROGA's load address (26002)

R1+34\B

- set breakpoint at an offset of 34 into PROGA (26036)

- proceed

R1+34(ADA R1+63) 1 115004 160010 114011 2 [CR]

R1+52/ 3 7 [CR]

- display and change "J" to 7

R1+52/ 7 [CR]

- display "J"

R1/ 0 [LF]

- display "IARY(1)"

R1+1/ 0 2\ P

- display "IARY(2)" and break after 2 breakpoint instructions

R1+34(ADA R1+63) 3 115004 160010 114011 2 [CR]

R1/ 1 7 [LF]

- change "IARY(1)" to 7

R1+1/ 2 [CR]

- display "IARY(2)"

26002/ 7 CD" [LF]

- change "IARY(1)" to "CD"

R1+1/ 2 [CR]

- display "IARY(2)"

R1+21/ LDA R1+53

CLA,INA [LF]

- change to CLA,INA instruction

R1+22/ STA R1+52

STA R1+54 [CR]

- change to store into "JJ"

R1+21/ CLA,INA

- verify change

# DBUGR COMMANDS

- ^      Print and open the previous location
- \* / or . /      Print and open the current location
- 0 / or 1 /      Print and open the A or B register
- \ M + 1 /      Print and open E and O registers
- \ M + 3 /      Print and open X register
- \ M + 4 /      Print and open Y register

# EXAMPLES

R1 1/ 2 [LF] — display "IARY (2)"

R1 + 2/ 0 3 [LF] — set "IARY (3)" to 3

R1 + 3/ 0 4 [CR] — set "IARY (4)" to 4

^ — display previous location

R1 + 2/ 3 ^ — "IARY (3)" and previous location

R1 + 1/ 2 [CR] — "IARY (2)"

1/ JSB 1004,I JSB 1003 [CR] — display and change B register

./ JSB 1003 [CR] — display current location

\M+4/ JSB 11,I — display Y register

# COMMANDS

## (PRINT MODE)

- `\S` Set print mode to symbolic instruction (default)
- `!` Print the last quantity typed as an instruction
- `\C` Set mode to constant
- `=` Print last quantity as a constant
- `\H` Set mode to ASCII characters
- `'` Print last quantity as two ASCII characters
- `n\R` Change the output radix to n



# EXAMPLES

## (PRINT MODE)

<u>\</u> M 3/ LDA 10,I [CR]	— display X register
<u>=</u> 160010 [CR]	— display X register as a constant
<u>R1+54/</u> ADA 502 [CR]	— display “JJ”
<u>'</u> AB” [CR]	— display “JJ” in ASCII
<u>\</u> C	— change mode to constant
<u>R1 54/</u> 40502 <u>'</u> AB” <u>!</u> ADA 502	— display “JJ” all three modes where 40502 octal = “AB” = ADA 502.
<u>16 =</u> 16 <u>16. =</u> 20 [CR]	— convert 20. to octal
<u>10.\</u> R	— set output radix to 10
<u>16 =</u> 14. <u>16. =</u> 16. [CR]	— convert 16 to decimal
<u>555 =</u> 365.	— convert octal to decimal

# CMM4

```

INPUT  FUNCTION
ID    LIST ID SEGMENT
EQ    LIST EQT AND EXTENTS
DR    LIST DEV REF TABLE
LM    LIST MEMORY
XL    LIST MEMORY    (SYSTEM MAP)
IN    LIST INTERRUPT TABLE
TA    LIST TRACK ASSIGNMENT TABLE
TR    TRACE LIST
XT    TRACE LIST (SYSTEM MAP)
LP    LIST DISC RES PROGRAM
DP    DISPLAY INPUT IN OCTAL DECIMAL & ASCII
PG    LIST ANY LOCATION IN PHYS MEMORY
PP    MODIFY ANY LOCATION IN PHYSICAL MEMORY
LL    CHANGE LIST DEVICE
PM    PATCH MEMORY
XP    PATCH MEMORY    (SYSTEM MAP)
F/    FIND A VALUE IN MEMORY
XF    FIND A VALUE IN MEMORY    (SYSTEM MAP)
LI    LIST ENTRY POINT
DI    REPORT DISC DICTIONARY ADDRESS OF  AN ENTRY POINT
LE    LIST ALL ENTRY POINTS IN SYS
OL    LIST DISC SECTOR
DM    DISC MOD  ANY LU
DS    DISC SEARCH
MS    MOVES DISC SFCTORS TO ANOTHER DISC AREA
NS    SET # OF SECTRS PER TRACK
FP    DISPLAY PAST DISK MOOS
/E OR EN OR EX TO EXIT
FOR MORE INFO DO A ??,INPUT
A PK AFTER THE INPUT GIVES A PACKED LISTING

```

ID, PROGRAM NAME  
 ID, SEGMENT NAME  
 ID, NUMBR \* ALL ID'S IN SYSTEM  
 OR USE IDPK,  
 EQ, NUMBR  
 EQ, NUMBR, NUMBR GIVES EQTS INCLUSIVE  
 OR USE EQPK,  
 DR, NUMBR  
 DR, NUMBR, NUMBR GIVES DRT ENTRIES INCLUSIVE  
 OR USE DRPK,  
 LM, ADDRESS  
 LM, ADDRESS, # OF WORDS  
 OR USE LMPK,  
 XL, ADDRESS (SYSTEM MAP)  
 XL, ADDRESS, # OF WORDS (SYSTEM MAP)  
 OR USE XLPK,  
 IN, NUMBR  
 IN, NUMBR, NUMBR GIVES INT TABLE ENTRIES INCLUSIVE  
 OR USE INPK,  
 TA  
 TA, LU \*  
 TA, LU \*, TRK #, # OF TRKS  
 OR USE TAPK,  
 TR, START LOCATION, LIST DELIMITER  
 XT, START LOCATION, LIST DELIMITER  
 LP, PROG NAME, REL ADDRESS  
 OR USE LPPK,  
 DP, VALUE  
 DP, VALUE, \*, VALUE  
 DP, VALUE, /, VALUE  
 DP, VALUE, +, VALUE  
 DP, VALUE, -, VALUE  
 PG, PG#, OFFSET, # OF WORDS  
 OR USE PGPK,  
 PP, PG#, OFFSET, NEW VALUE  
 OR USE PPPK,  
 LL, LIST LU#  
 PM, ADDRESS, NEW VALUE  
 XP, ADDRESS, VALUE (SYSTEM MAP)  
 F/, VALUE TO FIND, START ADDRESS, # OF WORDS  
 XF, VALUE TO FIND, START ADDRESS, # OF WORDS  
 LI, ENTRY POINT NAME  
 DI, ENTRY POINT NAME  
 LE  
 DL, LU, TRK, SECTR, # OF SECTORS  
 OR USE DLPK,  
 DM DISC MOD <INTERACTIVE>  
 DS, LU, TRK, VALUE TO FIND  
 SOURCE IS: DESTINATION IS:  
 MS, LU, TRK, SECTR, LU, TRK, SECTR, # OF SECTRS



## OPERATOR REQUESTS



\*ON,XYZ

ENVIRONMENT BEFORE A KEY IS PRESSED ON THE SYSTEM CONSOLE

1. System console is in input mode.
2. System console select code (SC) is 15.
3. System console driver is DVR05.
4. RTE is idle (no executing programs).

# OPERATOR COMMAND PROCESSING

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
\$CIC(RTIOC)		Entered by depressing a key on the console, since DVR05 (and DVR00) leaves each terminal with its device control flip-flop set. Interrupt causes instruction in trap cell location 15 to be executed (JSB \$CIC,I). Save machine state, turn off interrupts, etc. Use interrupt and EQT tables to find the driver continuation/completion address.

## DISCUSS TRAP CELLS & INTERRUPT TABLE

C.05(DVR05)	Set "OPATN" to tell system that operator wants service.
-------------	---

\$CIC \$TYPE(SCHED)	If console is not busy (OPFLG):
\$XSIO(RTIOC) \$XSIO	Request DVR05 to output "*". Request DVR05 to read operator input and return to TYP10 upon completion.

\$XEQ(DISP)	RTE idle loop (or resume execution of the interrupted programs).
-------------	--

Prompt ("\*") is output  
Operator inputs: "ON,XYZ"

\$CIC C.05(DVR05)	Save machine state, etc. Detect end of input line and completion of I/O request.
----------------------	---

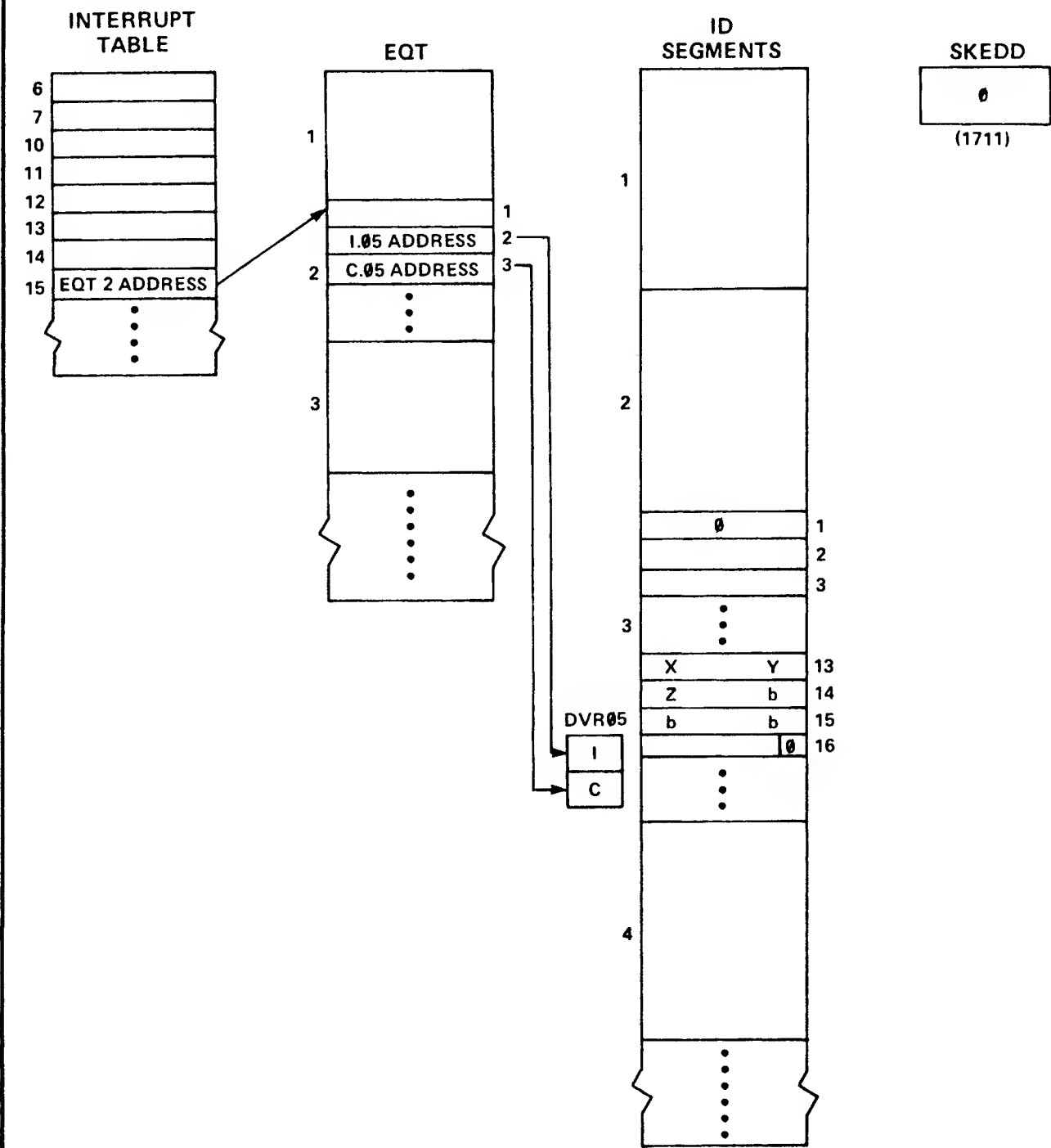
IOCOM (\$CON1)	Remove I/O request and jump to completion address.
----------------	--

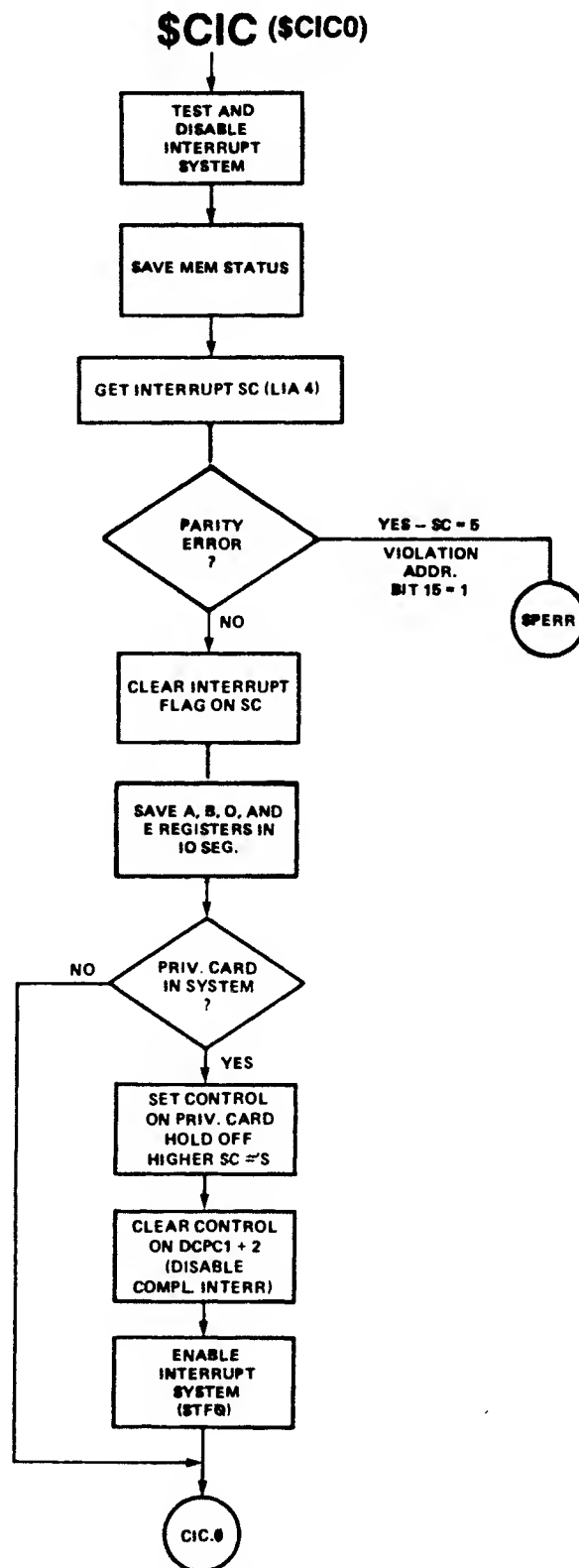


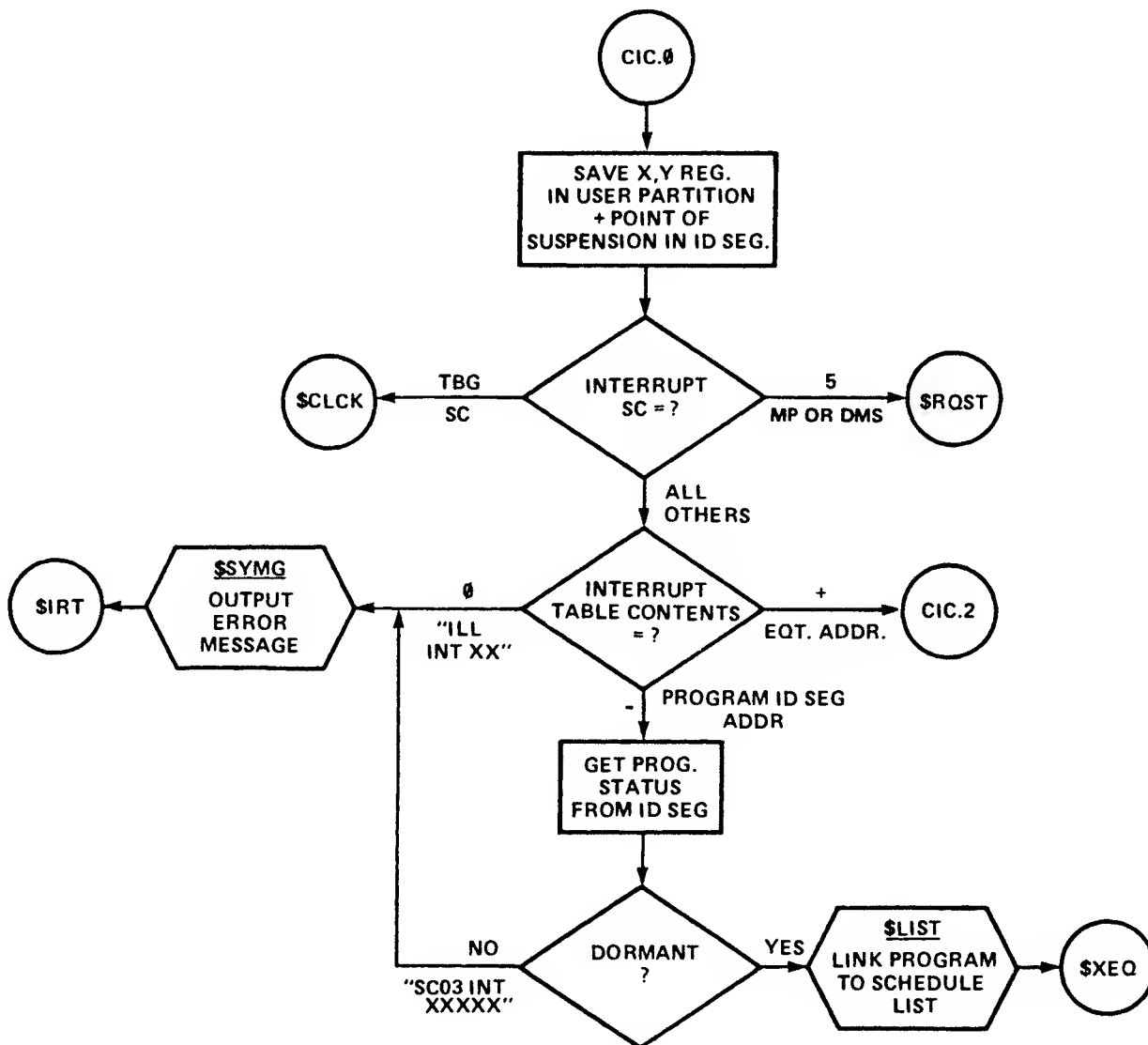
EXECUTION	SUBROUTINE CALLS	NOTES
-----	-----	-----
TYP10	\$MESS(\$MSG- SCHED)	Clear "OPFLG" (system console not busy) Operator command processor
	\$PRSE	Parse the request "ON,XYZ"
	DISCUSS KEYWORD BLOCK	
	M0100	ON,XXXX processor
	TTNAP	Find ID segment address of XYZ
	M0100	See if program is dormant
	PLOAD	Store parameters (P1-P5) into ID segment.
	\$LIST(SCHED)	Schedule* the program by changing its state and adding to the scheduled list.
TYP10		
	\$XSIO	Output message if necessary
\$XEQ		See if any programs are ready to be executed.
	X0010	Dispatch* program XYZ for execution.

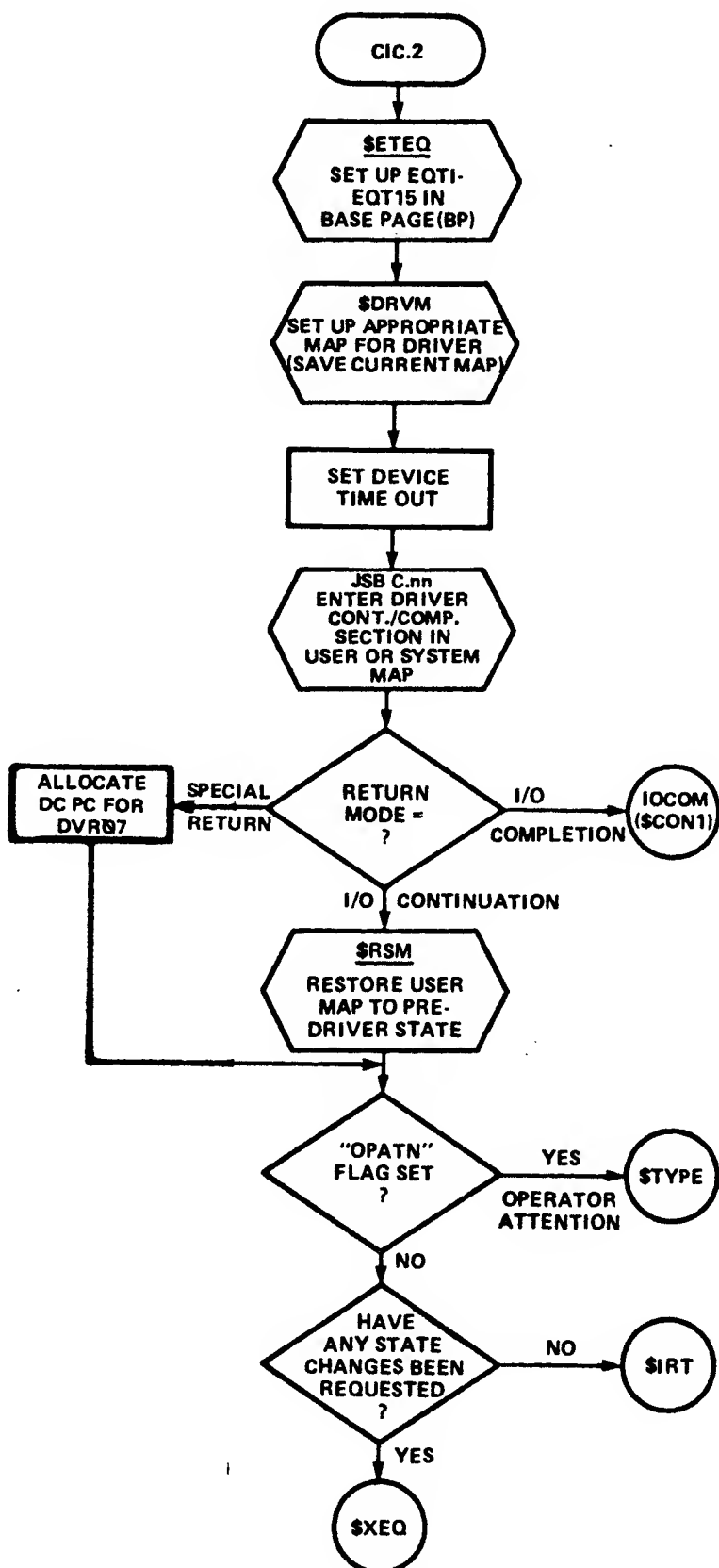
\*Note the distinction made in RTE between scheduling and dispatching a program.

# “ON,XYZ” TABLES/LISTS (INITIAL CONDITIONS)









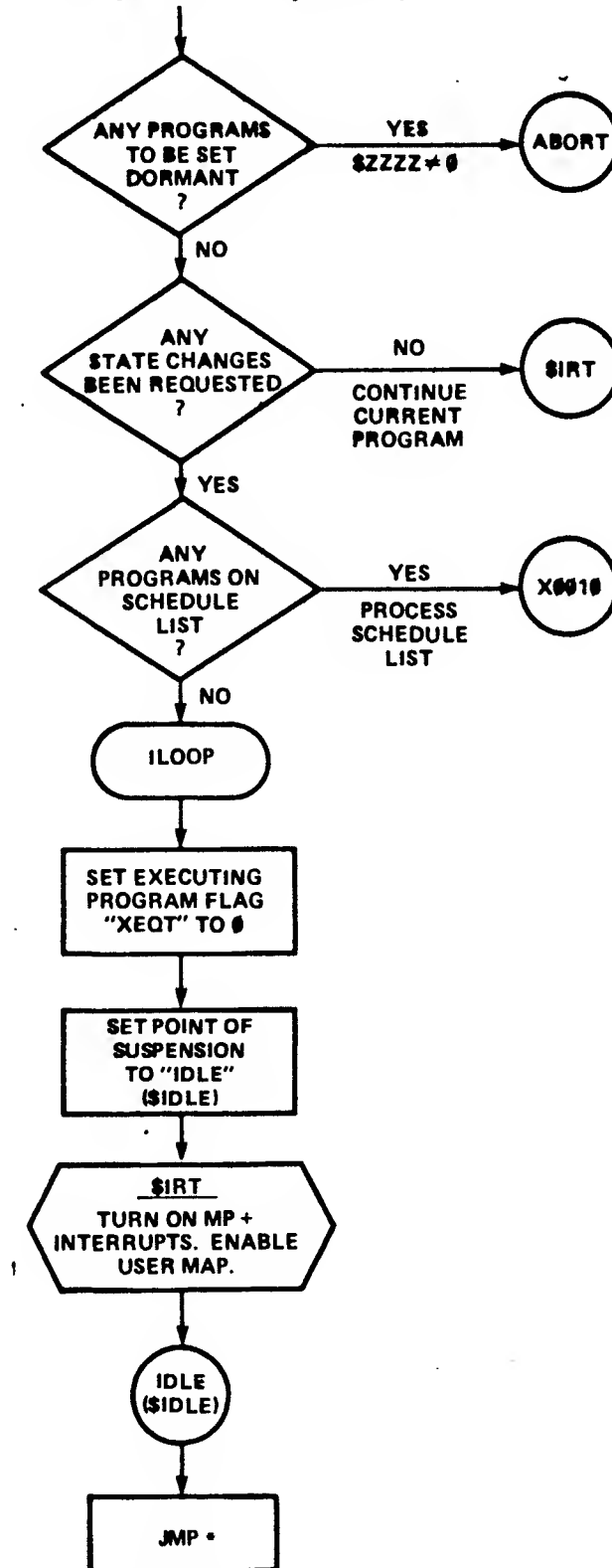
# ON, XXXXX COMMAND MESSAGE PROCESSOR

```

1527*
1528*****
1529*
1530*      ON(IH),XXXXX
1531*      ON(IH),XXXXX.NOW
1532*      ON(IH),XXXXX,P1,....,P5
1533*      ON(IH),XXXXX.NOW,P1,....,P5
1534*
1535*      THE ON REQUEST FUNCTIONS AS FOLLOWS:
1536*      IF NO RESOLUTION CODE, THEN PROGRAM SCHEDULED.
1537*      IF -NOW- OPTION, THEN ENTER PROGRAM INTO TIME LIST
1538*      AND SET TIME VALUES TO CURRENT TIME PLUS 10 MSC
1539*      IF NOT ONE OF ABOVE, AND TIME VALUES ARE ZERO THEN
1540*      PROGRAM FUNCTIONS SAME AS -NOW- OPTION.
1541*      IF NOT ONE OF ABOVE, AND TIME VALUES ARE PRESENT,
1542*      THEN PROGRAM IS ADDED TO TIME LIST.
1543*      NOTE: 1) ALL THE ABOVE OPTIONS ALLOW PARAMETERS TO BE
1544*      PASSED TO THE PROGRAM. THESE MUST BE ASCII
1545*      DECIMAL NUMBERS WHICH ARE CONVERTED TO BINARY
1546*      AND STORED IN ID SEGMENT TEMP AREA. UPON
1547*      EXECUTION, THE 3 REGISTER WILL POINT TO TEMP.
1548*      UP TO 5 PARAMETERS MAY BE INPUT. IF NO PARA-
1549*      METERS ARE INPUT, THE TEMP AREA ARE ZEROS BUT
1550*      R REGISTER WILL STILL POINT TO TEMP. AREA
1551*      2) THE ABOVE OPTIONS WILL ALLOW THE ORIGINAL
1552*      SCHEDULING STRING TO BE SAVED (UNLESS 'IH'
1553*      IS SPECIFIED OR THERE ARE NO PARAMETERS).
1554*      THE SCHEDULED PROGRAM MAY RECOVER THIS STRING
1555*      WITH AN EXEC 14 CALL.
1556*
1557*****
1558*
1559 01164 0174042 M0100 JSR TTNAM      FIND ID SEGMENT ADDR
1560 01165 1666059      LDB WSTAT,I   IF NO PARAMETERS
1561 01166 005222      RPL,RPL        BIT IS SET. THEN
1562 01167 006021      SSH,PSS        ILLEGAL STATUS.
1563 01170 002002      SZA            CHECK IF PROGRAM DORMANT
1564 01171 0274702      JMP M0405     ILLEGAL STATUS ERROR
1565 01172 0175502      JSR $57IT     CHECK OUT THE PROGRAM SIZE
1566 01173 002002      SZA            IS IT OK ?
1567 01174 0266702      JMP MSFX      NO, FLUSH HIM !
1568*
1569 01175 017267P      JSR FLOAD      GO TO PROCESS CONTROL PRAMETERS
1570 01176 066057X      LDB WORK
1571 01177 046026P      ADB D17        COMPUTE RES/T/MULT ADDR
1572 01200 169001      LDA R,T
1573 01201 001723      ALF,PAR
1574 01202 012022P      AND D7        CHECK RESOLUTION CODE
1575 01203 002002      SZA            NONE, SO GO TO SCHED NOW
1576 01204 027210P      JMP M0110
1577 01205 016032X M0105 JSR $1TST     SCHEDULE PROGRAM
1578 01206 000301      OCT 301
1579 01207 026670P      JMP MSFX      RETURN
1580 01210 006004 M0110 INB            SET R FOR $ONTM
1581 01211 062036P      LDA CP2        IF ASCII

```

**\$XEQ (\$XCQ)**



# OPERATOR COMMAND PROCESSORS

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
RT	M0070	\$OTRL \$SDRL	SETS CALL TO \$SDRL SCANS TAT, RELEASES TRACKS
ON	M0100	\$LIST \$ONTM \$TADD	IF NO TIME PRAMS THEN SCHEDULE PROGRAM, IF AVAILABLE. IF TIME PRAMS THEN SET START TIME ADDS PROG TO TIME LIST
OF(,0)	M0200	SABRT \$TREM \$LIST	SOFT ABORT REMOVE FROM TIME LIST SET DORMANT
OF(,1) or OF(,8)		\$IOCL \$ABRT \$TREM \$SDRL \$SYMG	IF I/O SUSP, CLEAR I/O HARD ABORT(OF,1) REMOVE FROM TIME LIST RELEASE DISC TRACKS ABORT MESSAGE
		\$DREL	RELEASE PROG'S TRACKS (OF,8 ONLY)
SS	M0300	\$LIST	PUT IN SUSPEND LIST
GO	M0400	\$ILST \$LIST	ILLEGAL STATUS MESSAGE OR PUT IN SCHEDULED LIST
ST	M0500	\$CNVL	FORMAT MESSAGE PARAMETERS
PR	M0650	\$INER \$LIST	ERROR MESSAGES RE-LINK THE PROGRAM
IT	M0600	\$INER \$ETTM \$TREM	ERROR MESSAGES SET TIME VALUES IN ID SEG REMOVE FROM T-LIST IF RES =0
TM	M0700	\$TMRQ \$INER \$ETTM	DOES ALL THE WORK ERROR MESSAGES SETS TIME VALUES IN CLOCK
DN	M0800 \$IODN	\$IODN \$INER \$CVEQ \$LIST	DOES THE WORK ERROR MESSAGES GET EQT ADDRESS WAIT LIST FOR PROGS USING DEV.



# OPERATOR COMMAND PROCESSORS (cont'd)

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
UP	\$IOUP	\$INER \$CVEQ \$SCD3 \$LIST XUPIO \$XXUP LINK DRIVR \$SYMG	ERRORS GETS EQT ADDRESS RE-SCHEDULE WAITING PROGS PUT IN SCED. LIST UP LU'S ASSOCIATED WITH THIS EQT LINK AN I/O QUEUE ONTO THIS EQT LINK AN I/O REQUEST ON THIS EQT INITIATE WAITING I/O DIAG. MESSAGE IF STILL N/A
LS	M0960	\$INER	ERRORS
LG	M0970	\$CREL \$DFEQ	RELEASE CURRENT TRACKS ALLOCATE NEW TRACKS
TI	M0750	\$TIMV \$CNV1	GETS TIME VALUES FORMATS INTO OUTPUT MESSAGE
BR	M0725		DOES ALL THE WORK
AB	M0950	\$ABPT \$TRE1 \$LIST	SOFT ABORT, ENTERS OF OR BR REMOVE FROM TIME LIST, REFER TO OF OR BR PROCESSING SET IN DOMANT LIST
RU	M0408	\$SZIT \$LIST	CHECK PROGRAM VS. MAX PARTITION SIZE. SCHEDULE PROGRAM
BL	BLIM	\$CNV1	FORMAT MESSAGE

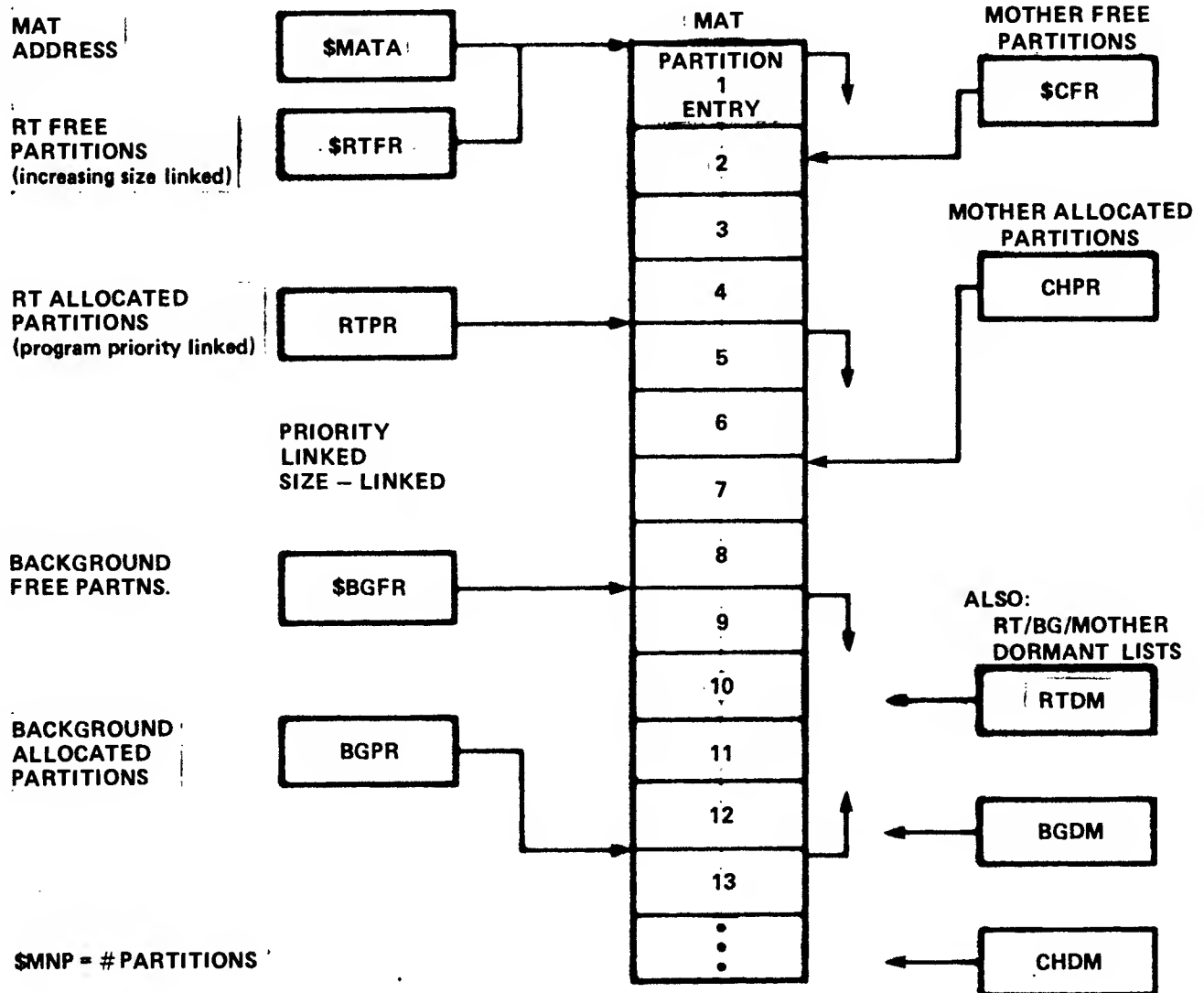
# OPERATOR COMMAND PROCESSORS (cont'd)

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
-----+-----			
LU	M0850	\$LUPR	--- PROCESSED IN OCMD4 MODULE       -----+
EQ	M0900	\$EQST	
TO	M0990	\$CHTO	
SZ	SIZE	SZCHK	CHECK SIZE CHANGE GET
		\$SZIT	GET PROGRAM SIZE PARAMETERS
AS	ASIGN	SZCHK	VERIFY ASSIGNMENT
		\$SZIT	GET PROGRAM SIZE PARAMETERS
UR	URES		DOES IT ALL

PROGRAM DISPATCHING  
PARTITION ASSIGNMENT



# MEMORY ALLOCATION TABLE (MAT)



(NOTE: listheads are kept in Table Area 2 and DISPM)

- last entry points to start of allocated list

- priority linked

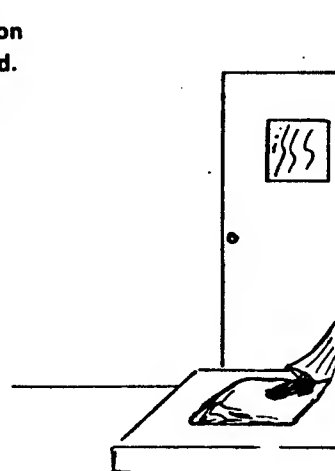
# MAT ENTRY

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Linkage pointer to next entry (-1 if undefined partition)															
1	Priority of current resident program															
2	Current resident's ID-segment address															
3	M		D					Beginning page of partition								
4	R	C						Number of pages in partition (-1)								
5	RT														S	
6	Subpartition Link Word (SLW)															

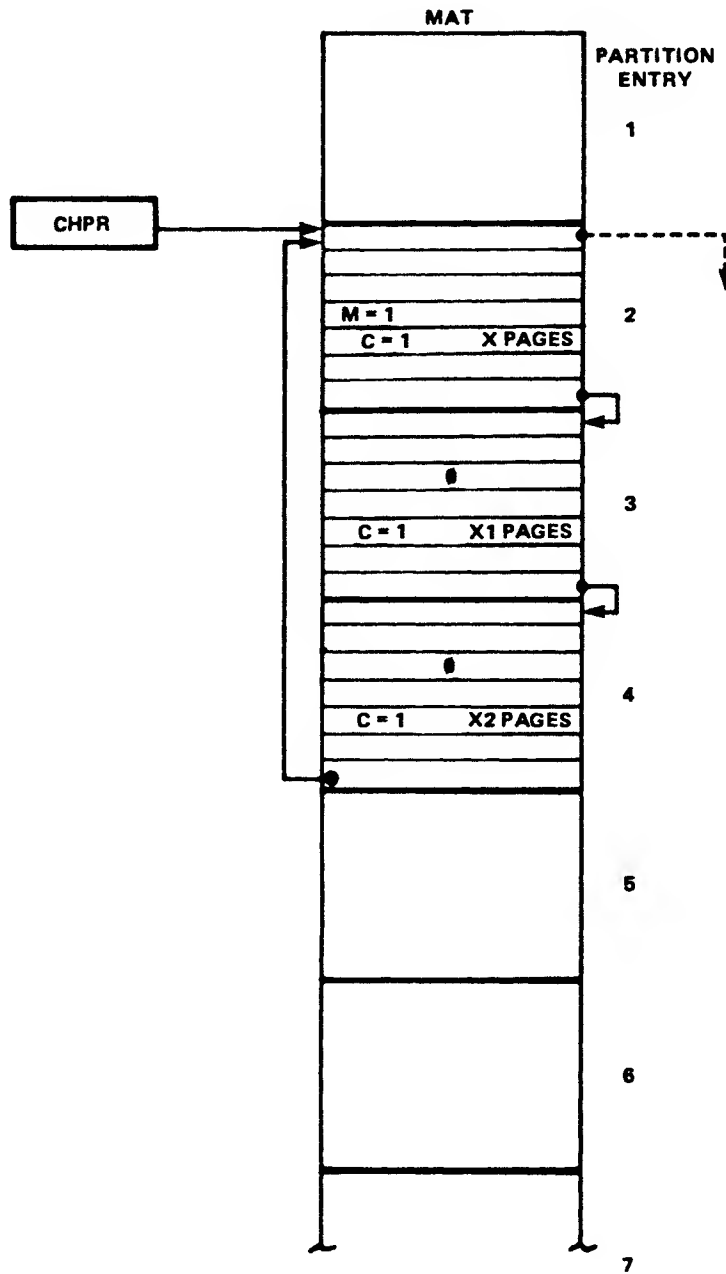
- D = Resident is dormant — save resources, serially reusable, or operator suspended
  - M = Mother partition
  - R = Partition is reserved
  - C = Partition is part of a chained mother partition
  - RT = REAL TIME PARTITION
  - S = Program's dispatching status
    - 0 - Read in progress
    - 1 - Program is resident
    - 2 - SWAP out or segment load in progress
    - 3 - SWAP out complete but program still resident
    - 4 - Subpartition swap-out started for mother partition
    - 5 - Subpartition swap-out completed. Mother cleared.
- "NORMAL" sequence: 2,3,0,1

## SLW

- = 0 - Partition not a subpartition
- = Next subpartition
- = Mother partition if partition is last subpartition

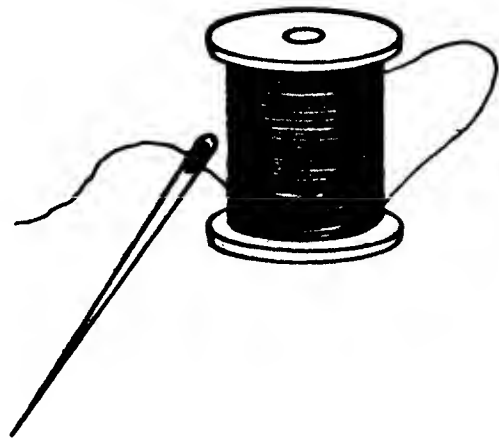
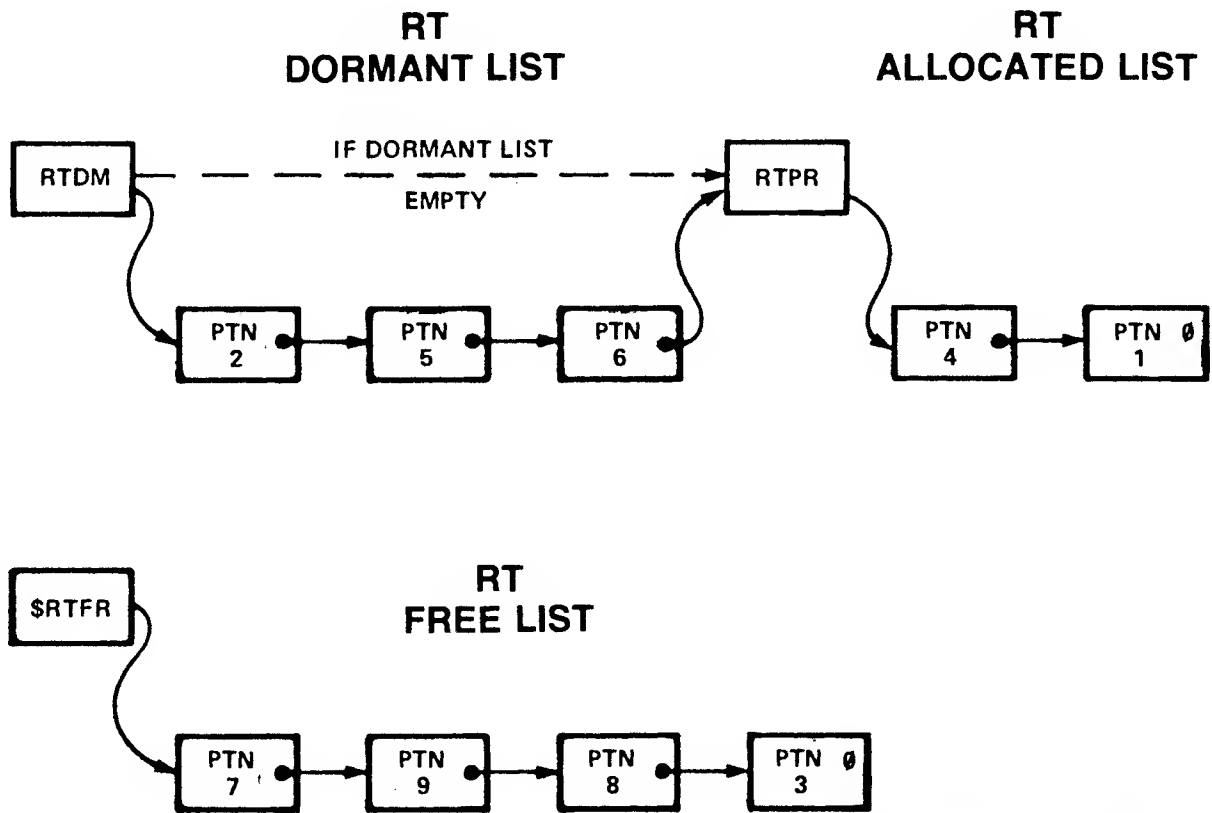


# **SAMPLE MOTHER/SUBPARTITION LIST THREADING IN MAT**



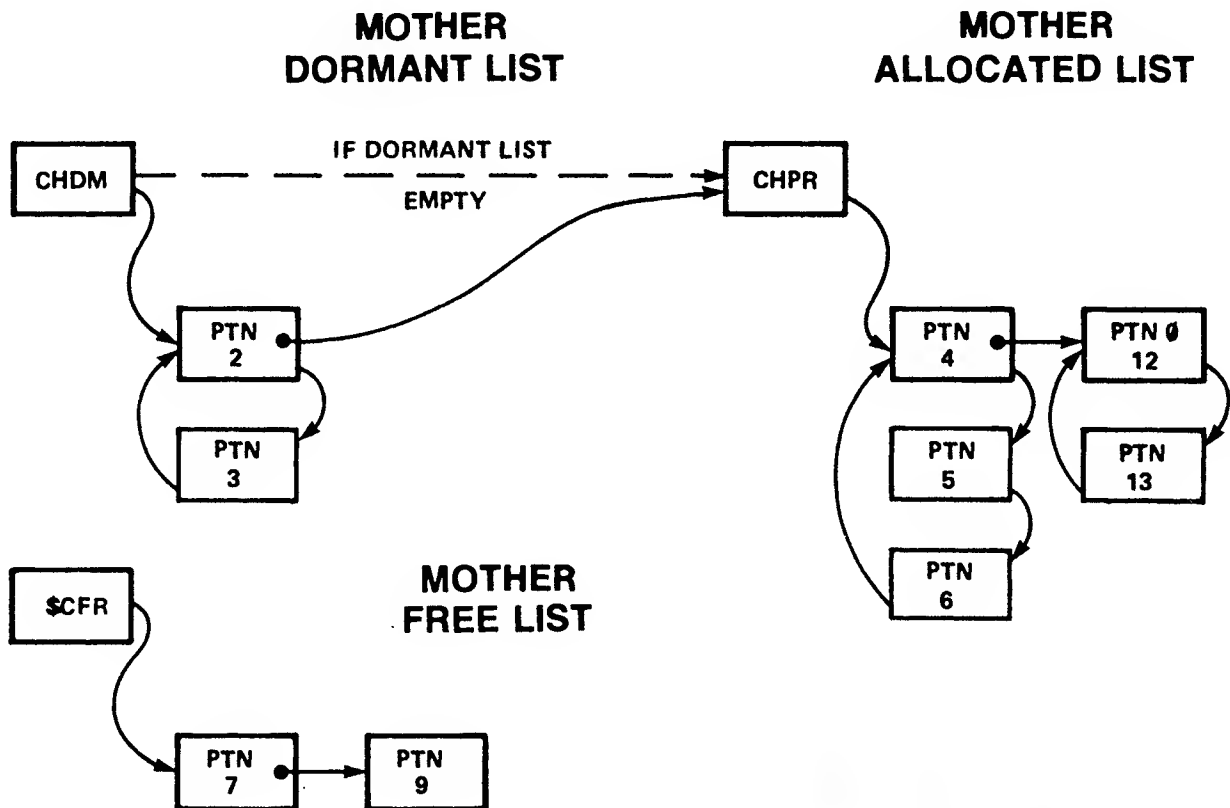
NOTE:  $X > X1 + X2$

# SAMPLE LIST THREADING IN MAT





# SAMPLE LIST THREADING IN MAT



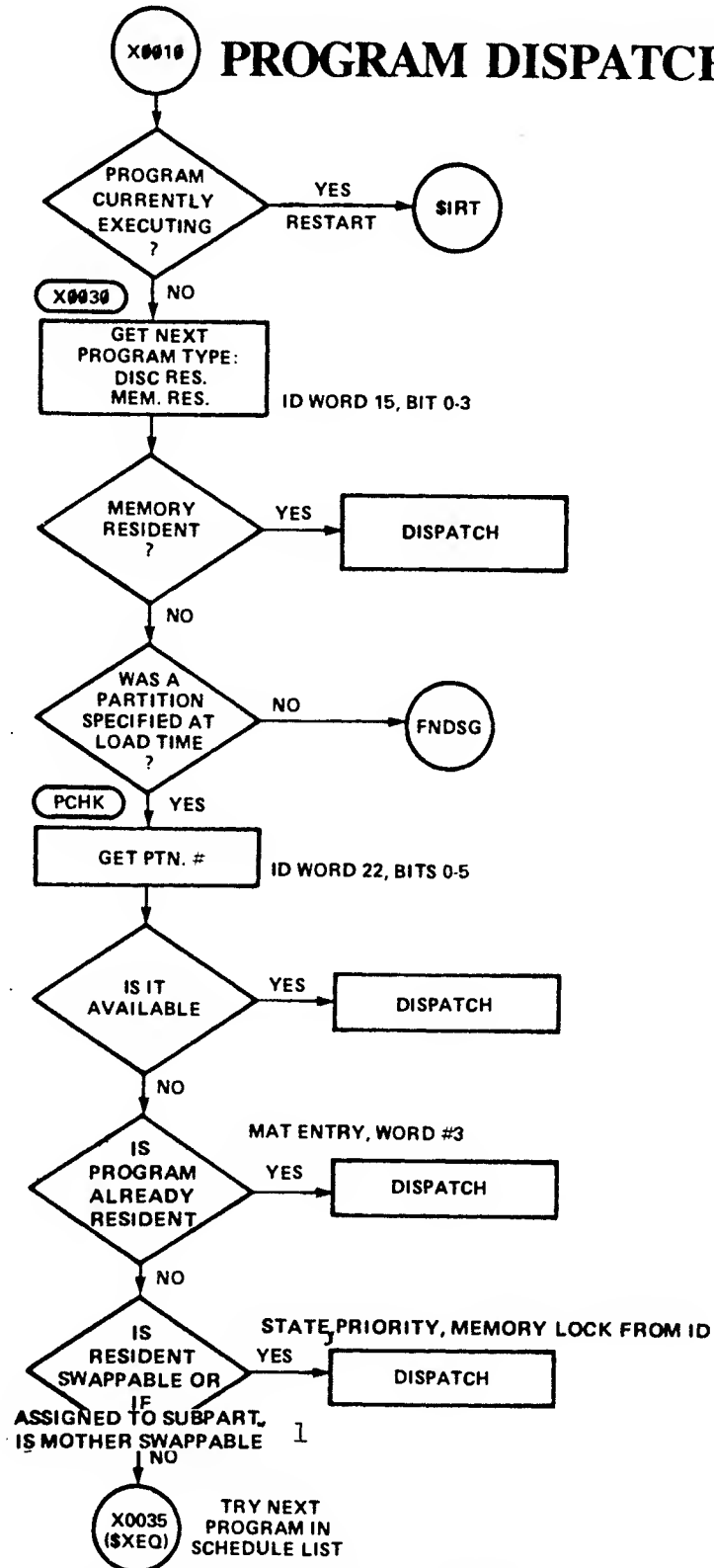
# MAT EXAMPLE

WORD	LOCATION	VALUE(8)			
1	24317	23500	—	\$MATA	
WORD	LOCATION	VALUE(8)			
1	24336	23534	—	\$BGFR	
2	24337	0	—	\$RTFR	
WORD	LOCATION	VALUE(8)			
1	24350	23525	—	\$CFR	
2	24351	17	—	\$MNP	
WORD	LOCATION	VALUE(8)			
1	27274	23552	—	BGDM	
2	27275	23500	—	BGPR	
3	27276	24336			
4	27277	27275			
5	27300	27274			
6	27301	27302	—	RTDM	
7	27302	0	—	RTPR	
8	27303	24350			
9	27304	27307			
10	27305	27306			
11	27306	27307	—	CHDM	
12	27307	0	—	CHPR	
WORD	LOCATION	VALUE(8)			
1	23500	23543			
2	23501	77777			
3	23502	17612			
4	23503	55			
5	23504	1			
6	23505	1			
7	23506	0			
8	23507	0			
9	23510	1			
10	23511	17551			
11	23512	57			
12	23513	4			
13	23514	1			
14	23515	0			
15	23516	23507			
16	23517	132			
17	23520	17714			
18	23521	64			
19	23522	12			
20	23523	1			
21	23524	0			
22	23525	0			
23	23526	62			
24	23527	0			
25	23530	100077			
26	23531	100			
27	23532	1			
28	23533	23534			
29	23534	23561			
30	23535	62			
31	23536	0			
32	23537	77			
33	23540	6			
34	23541	1			
35	23542	23543			
36	23543	23516			
37	23544	132			
38	23545	17141			
			39	23546	106
			40	23547	6
			41	23550	1
			42	23551	23552
			43	23552	27275
			44	23553	24
			45	23554	17202
			46	23555	20115
			47	23556	25
			48	23557	1
			49	23560	23561
			50	23561	0
			51	23562	0
			52	23563	0
			53	23564	143
			54	23565	34
			55	23566	0
			56	23567	23525
			57	23570	177777
			58	23571	0
			59	23572	0
			60	23573	0
			61	23574	0
			62	23575	0
			63	23576	0
			64	23577	177777
			65	23600	0
			66	23601	0
			67	23602	0
			68	23603	0
			69	23604	0
			70	23605	0
			71	23606	177777
			72	23607	0
			73	23610	0

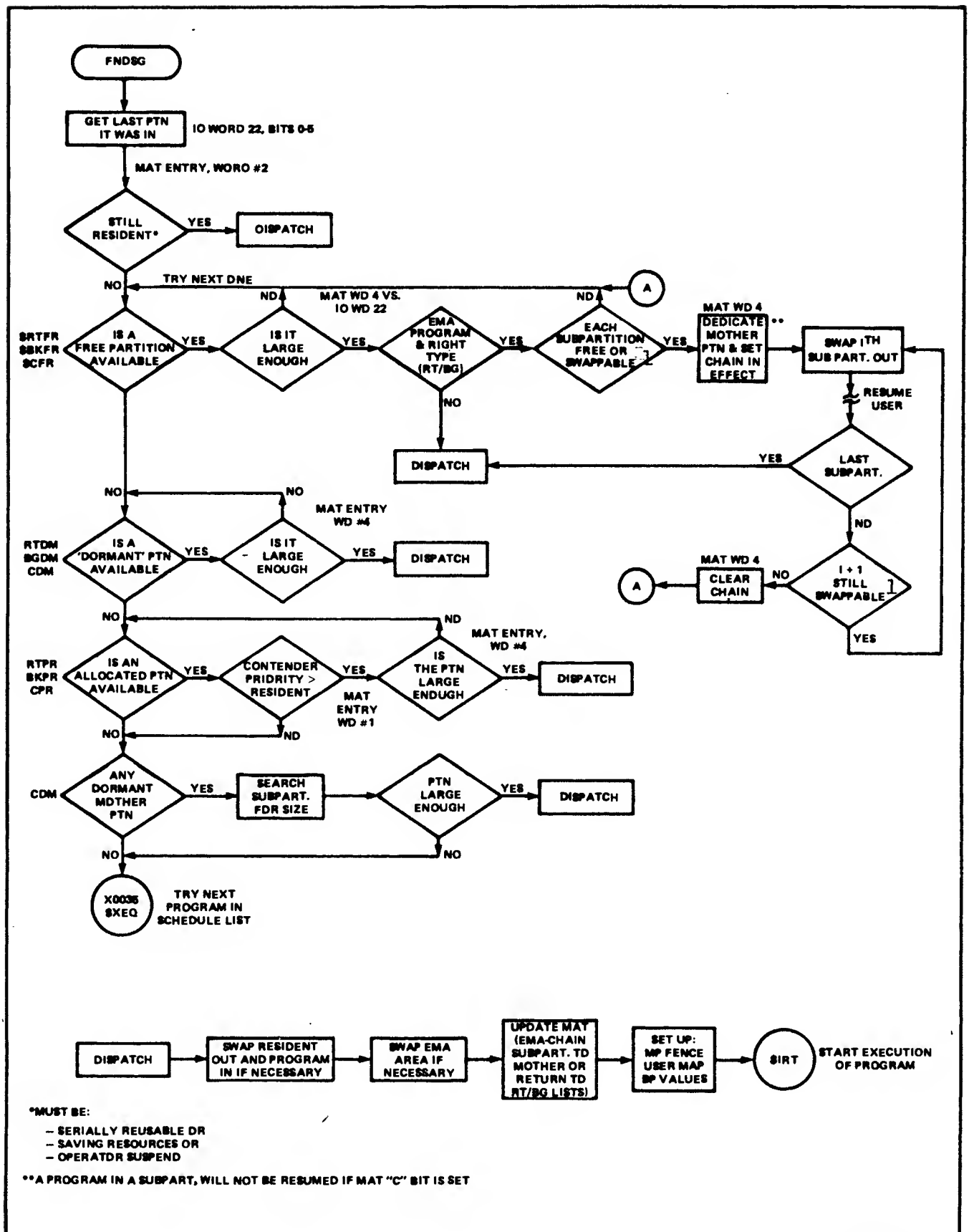
**DATA SAMPLE**  
(EMA PROGRAM RUNNING)

WORD	LOCATION	VALUE(8)	
1	24317	23500	\$MATA
WORD 1	24336	0	\$BGFR
2	24337	0	\$RTFR
WORD 1	24350	0	\$CRF
2	24351	17	\$MNP
WORD 1	27274	27275	BGDM
2	27275	23500	BGPR
3	27276	24336	
4	27277	27275	
5	27300	27274	
6	27301	27302	RTDM
7	27302	0	RTPR
8	27303	24350	
9	27304	27307	
10	27305	27306	
11	27306	27307	CHDM
12	27307	23525	CHPR
WORD 1	23500	23516	
2	23501	77777	
3	23502	17612	
4	23503	55	
5	23504	1	
6	23505	1	
7	23506	0	
8	23507	0	
9	23510	1	
10	23511	17551	
11	23512	57	
12	23513	4	
13	23514	1	
14	23515	0	
15	23516	23507	
16	23517	132	
17	23520	17714	
18	23521	64	
19	23522	12	
20	23523	1	
21	23524	0	
22	23525	0	
23	23526	62	
24	23527	20016	
25	23530	100077	
26	23531	40100	
27	23532	1	
28	23533	23534	
29	23534	23507	
30	23535	132	
31	23536	0	
32	23537	77	
33	23540	40000	
34	23541	0	
35	23542	23543	
36	23543	27275	
37	23544	24	
38	23545	0	

39	23546	20100
40	23547	40000
41	23550	0
42	23551	23552
43	23552	23561
44	23553	120
45	23554	0
46	23555	115
47	23556	40025
48	23557	0
49	23560	23561
50	23561	0
51	23562	0
52	23563	0
53	23564	143
54	23565	40034
55	23566	0
56	23567	23525
57	23570	177777
58	23571	0
59	23572	0
60	23573	0
61	23574	0
62	23575	0
63	23576	0
64	23577	177777
65	23600	0
66	23601	0
67	23602	0
68	23603	0
69	23604	0
70	23605	0
71	23606	177777
72	23607	0



1. SWAPPABILITY IS DETERMINED BY STATE, PRIORITY, MEMORY LOCK FROM ID, &SWAP DELAY.





I/O  
PROCESSING





## I/O PROCESSING

### GENERAL OPERATION

- I/O operations are performed concurrently with program computation.
- I/O transfer can be broken into three phases:
  - Initiation
  - Continuation
  - Completion
- User programs are involved only in the initiation and completion phases.
- I/O request types include:
  - User Normal Operation
  - User Automatic Buffering
  - User Class I/O
  - System Requests
- I/O drivers operate under control of IOC and SCIC system modules.
- I/O drivers are composed of two sections:
  - Initiation
  - Continuation/Completion
- \$XSIO handles all I/O requests from the system modules.
- EQTs associate drivers with devices.

## I/O INITIATION

- User program makes an EXEC call to initiate I/O transfers.
- Request specifies LU, buffer location, buffer length, and request type (read, write, or control).
- IOC calls initiation section of driver to start the data transfer.
- User program will be suspended or restarted depending on the I/O request type.

## I/O CONTINUATION

- Device will interrupt after completion of first data transfer
- CIC calls the continuation/completion section of the driver.
- The driver completes the first transfer and starts the next data transfer.
- The currently executing program is restarted.

## I/O COMPLETION

- After the last interrupt the driver notifies \$CIC that the I/O transfer has completed.
- IOCOM is called to terminate the I/O request and initiate the next transfer if any requests are stacked on the EQT.
- The suspended user program is then placed on the scheduled list.



## I/O REQUEST STACKING

-----

Requests are stacked in lists for each device according to priority. The formats of the four types of requests as they appear in the I/O lists are:

### 1. USER (NORMAL OPERATION)

-----

The parameters from the request are stored in the temporary area of the program ID segment. The link word of the segment is used to link into the I/O list.

WORD	CONTENTS
----	-----
1	< LINKAGE WORD >
2	<T, CONTROL INFO, REQUEST CODE >
3	<L, BUFFER ADDRESS (L=1 IF IN SAM) >
4	<BUFFER LENGTH >
5	<DISC TRACK ADDR OR ZERO >
6	<DISC SECTOR ADDR OR ZERO >
7	<PROGRAM PRIORITY >
.	-REMAINDER OF ID SEGMENT .

### 2. USER (AUTOMATIC OUTPUT BUFFERING)

-----

Requests of this type are constructed in system available memory.

WORD	CONTENTS
----	-----
1	<LINKAGE WORD >
2	<T, CONTROL INFO, REQUEST CODE>
3	<PRIORITY OF REQUESTOR >
4	<TOTAL BLOCK LENGTH WORDS >
5	<USER BUFFER LENGTH >
6	<OPTIONAL PARAMETER 1 >
7	<OPTIONAL PARAMETER 2 >
8	<WORD 1 OF USER BUFFER >
.	.
.	.
N+7	<WORD N OF USER BUFFER >

### 3. USER (CLASS INPUT/OUTPUT)

-----

Requests of this type are constructed in system available memory.

WORD	CONTENTS
----	-----
1	<LINKAGE WORD >
2	<T, CONTROL INFO, REQUEST CODE>
3	<PRIORITY OF REQUESTOR > (CHANGED TO STATUS AT COMP.)
4	<TOTAL BLOCK LENGTH WORDS >
5	<CLASS ID WORD >
6	<USER BUFFER LENGTH > (CHANGED TO TLOG AT COMP.)
7	<OPTIONAL PARAMETER 1 >
8	<OPTIONAL PARAMETER 2 >
9	<WORD 1 OF USER BUFFER >
.	.
.	.
N+8	<WORD N OF USER BUFFER >

### 4. SYSTEM REQUEST

-----

The system request is linked into the I/O list by using word 4 of the call as a link word. A system request assumes the priority level of zero (highest priority).

WORD	CONTENTS
----	-----
1	< USB \$XSIO >
2	< LOGICAL UNIT # >
3	<COMPLETION ROUTINE ADDR >
4	< LINKAGE WORD >
5	<T, CONTROL INFO, REQUEST CODE>
6	<BUFFER ADDR OR DISC CNTL >
7	<BUFFER LENGTH OR PRIORITY >
8	<MAP WORD WHERE: >

BIT 15	BITS 0-14	SYSTEM MAP
-----	-----	-----
0	0	SYSTEM MAP
0	ID SEG.	USER MAP
1	0	CURRENT USER MAP
1	ID SEG.	MODIFIED USER MAP (I.E. EMA SWAP)

<T> FIELD:

-----

The <T> field (bits 15-14 in control word) identifies the request type as:

00 USER (NORMAL OPERATION)  
01 USER (AUTOMATIC BUFFERING)  
10 SYSTEM  
11 CLASS I/O

<REQUEST CODE> FIELD:

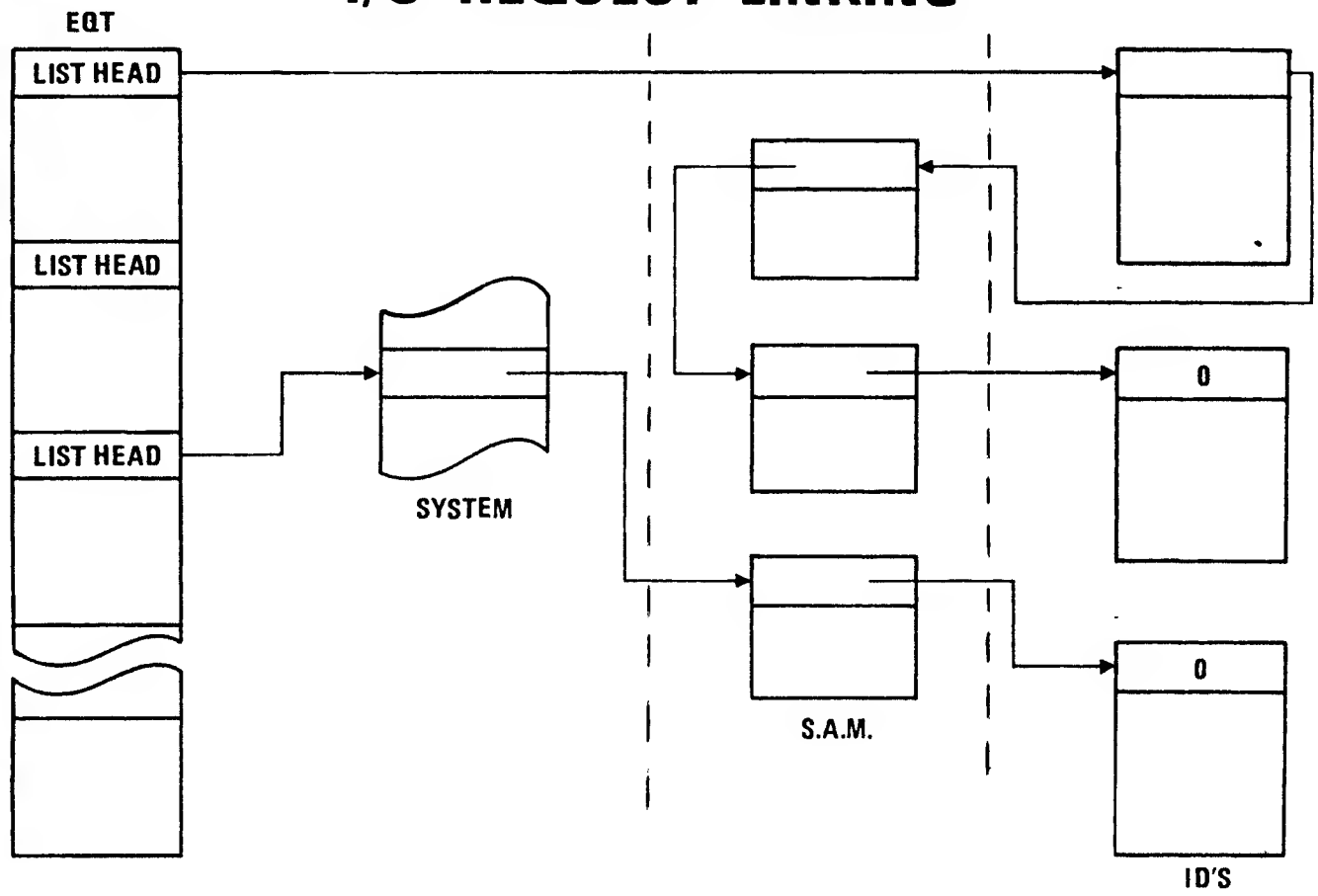
-----

1 READ  
2 WRITE  
3 CONTROL

<CONTROL INFO> FIELD INCLUDES SUBCHANNEL NUMBER



# I/O REQUEST LINKING



SYSTEM I/O REQUEST PROCESSOR  
-XSIO-

- XSIO allows RTE modules to call for I/O operations without the overhead of a user I/O request.
- Error checking is not performed
- System request has priority 0
- System disc call can specify a series of transfers
- A completion routine can be specified

EXEC CALL PROCESSING



CALL EXEC(2,2,IBUF,LEN,50,0)

```
JSB EXEC
DEF RET      Return address
DEF N2       Request code
DEF N2       Disk LU
DEF IBUF     Output buffer
DEF LEN      Buffer length
DEF N50      Track
DEF ZERO     Sector
RET EQU *
```

#### ENVIRONMENT

1. Calling program previously allocated track 50.
2. Disc (unbuffered) has no other requests in progress or stacked.
3. Disc is a 7905 using DVR32 on select code 11.

## EXEC 2 PROCESSING

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
MEMORY PROTECT		Caused by JSB EXEC in user program
TRAP CELL 5(JSB \$CIC,I)		MP violation generates an interrupt on select code 5.
\$CIC(\$TB1)		Enable system map and jump to \$CIC (\$CIC0).
\$CIC0(RTI0C)		Save machine state, turn off interrupts, etc. Detect MP violation (CIR=5).
\$RQST(EXEC)		Save violation address in program's ID segment.
		<div style="border: 1px solid black; padding: 5px; margin: 10px 0;">DISCUSS ID SEGMENT</div> <p>Detect MP error and not DMS violation</p> <p>Check that violating instruction was a JSB or JSB,I.</p> <p>Verify that destination address was "EXEC".</p>
RU(\$RQST)		<p>Check return address and number of parameters in request. (1&lt;Pn&lt;9)</p> <p>Store addresses of request parameters in BP.</p> <p>Verify request code and that address of each parameter to be used for storage is above MP fence.</p> <p>Jump to the request processor.</p>
\$IORQ(RTI0C)		<p>Verify LU number.</p> <p>Get EQT entry number from DPT.</p>

DISCUSS DRT

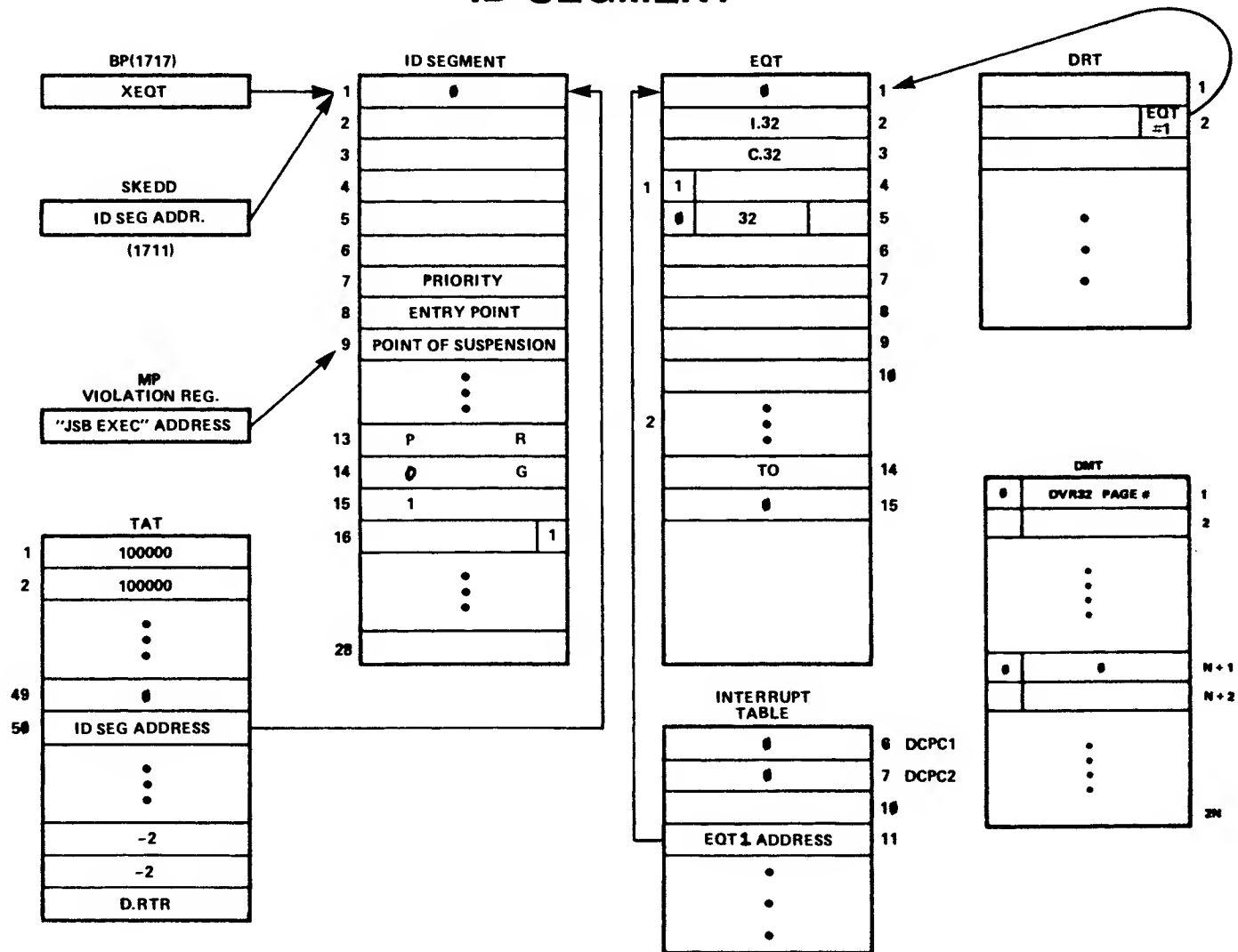
EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
	SCVEQ STADV	Transfer EQT entry addresses into BP. Ensure that EQT entry and LU are both up. If not suspend the program (state 3) by calling \$LIST.
L.01(\$IORQ)		Check that disc request has 5 request parameters. Verify track and sector numbers. Check TAT to insure that the user program owns track 50.
	DISCUSS TAT	
L.10(\$IORQ)		Setup ID segment words 2 thru 6: 2 - T, control, code 3 - Buffer address 4 - Buffer length 5 - Track 6 - Sector
	\$LIST	Put the program in I/O suspend state (state 2)
L.13(\$IORQ)	LINK	Link ID segment into the EQT entry.
	DRIVR	Assign a DCPC channel to the device EQT.
	DRVMP	Set up appropriate map for driver (DVR32). In this example the current user map will be modified to include DVR32's driver partition. In general the system map is used when: - driver is in SDA and does own mapping - I/O request is a buffered, class, or system request (driver ptn. must be mapped.)  User map is used when: - driver is in SDA and program is type 3 - I/O request is unbuffered (driver ptn. must be mapped)
		Transfer request parameter into EQT entry (words 4-10, and 15).

DISCUSS DMT

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
	I.32(DVR32)	Enter driver initiator section under system or user map. Start DCPC data transfer.
	<div style="border: 1px solid black; padding: 5px; display: inline-block;">DISCUSS TRACK MAP TABLES</div>	
	DRIVR	Set EQT entry availability (AV) to busy (2).
	<div style="border: 1px solid black; padding: 5px; display: inline-block;">DISCUSS EQT</div>	
\$XEQ		Schedule next program.
DCPC CHANNEL		DCPC moves words between user program's buffer and disc track 50.
DEVICE COMPLETION INTERRUPT		
\$CIC(RTIOC)		Save machine state, etc.
	C.32(DVR32)	Detect end of data transfer.
IOCOM or \$CON1 (RTIOC)		Return DCPC channel and clear TO. Unlink request from EQT entry.
L.51(IOCOM)	\$LIST	Place program into schedule list. Set EQT entry availability (AV) to 0 (available).
L.68(IOCOM)	DRIVR	If EQT entry has I/O requests stacked initiate next request.
IOCX		Assign available DCPC channel. If more than one device is waiting for a channel, the order of priority for assignment is the order of the Positions in the Equipment Table. There are two exceptions to this scheme:  1. If the first entry in the EQT is waiting for a DCPC, the channel is assigned to that device, which is assumed to be the system disc.  2. If the first entry encountered (other than entry #1) just released a DCPC channel, then the next lower priority device waiting for DMA is used. This allows for a "switching" operation in the allocation of a DMA channel.
\$XEQ(DISPM)		Dispatch the next program.



# CALL EXEC 2 TABLES/LISTS (INITIAL CONDITIONS) ID SEGMENT



ASSUME THE DISC IS BUSY  
(A DISC I/O REQUEST IS IN PROGRESS)

In this case: EQT entry 1 word 5 (AV field) would be 2(busy).  
Word 1 would be the link address of the request being processed.

EXECUTION	SUBROUTINE CALLS	NOTES
-----	-----	-----
		Flow of the EXEC 2 call remains the same as previously discussed (pg. 10-2 thru 10-3) until:
L.13 (\$IORQ) \$XEQ	LINK	Link ID segment into EQT entry. Since the device is busy (EQT entry 1 word #0), schedule the next program.

Once the current request is completed and \$CIC has called IOCOM:

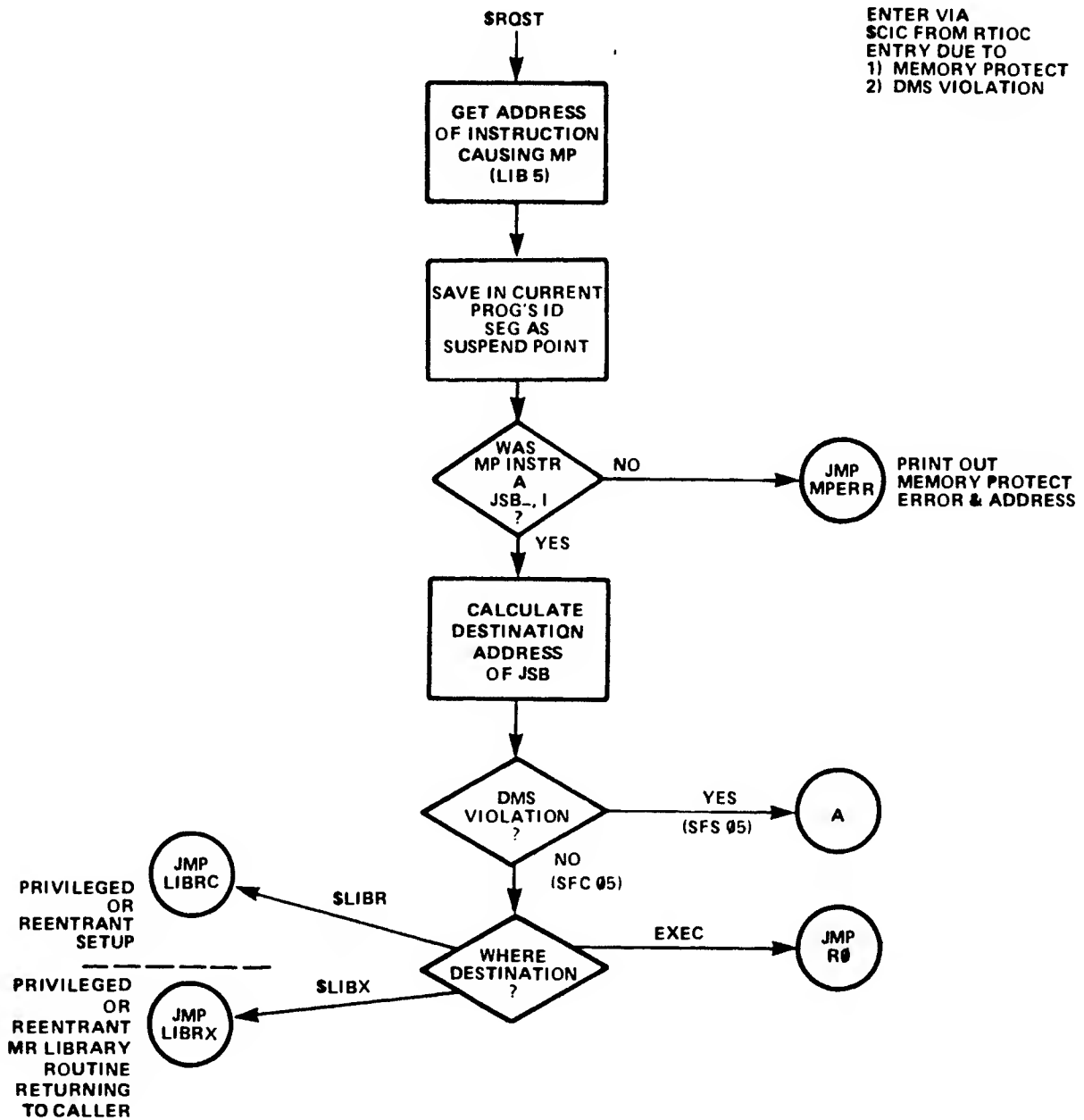
L.68 (IOCOM)	DRIVR	DRIVR is called since EQT entry 1, word 1 #0.
--------------	-------	---

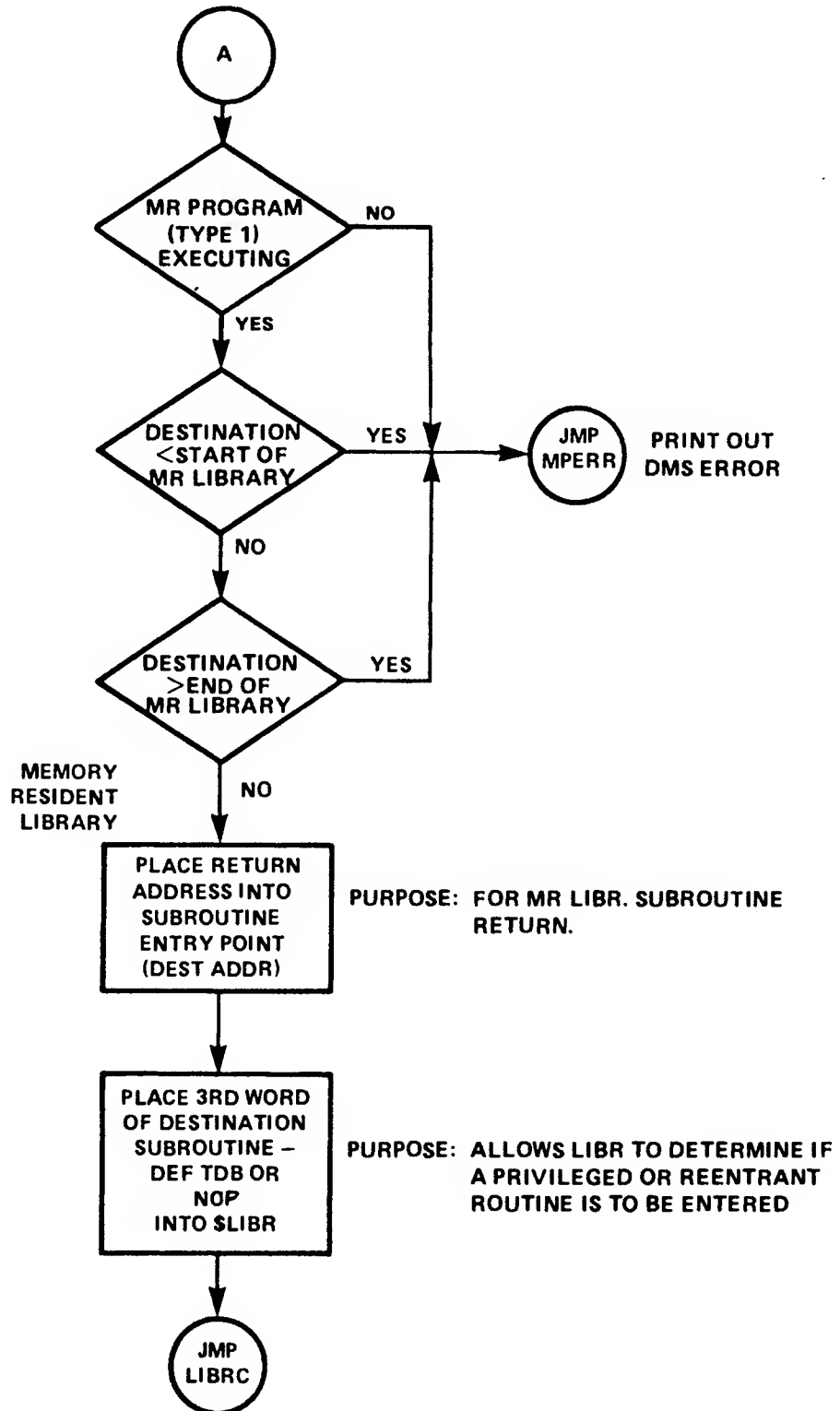
From this point on the flow is the same as the previous discussion following the first call to DRIVR from L.13 (pg. 10-3).

FLOW OF EXEC 2 REQUEST  
FOR A BUFFERED DEVICE  
(NON DISC DEVICE)

EXECUTION -----	SUBROUTINE CALL -----	NOTES -----
Flow is the same as the disc request example until (pg. 10-3):		
L.01(\$IORQ(RTIOC))		Detect that it is not a disc request (DVR30,31,32, or 33).
L.02(\$IORQ)		Check for LU lock.
L.027(\$IORQ)		Calculate SAM buffer size needed for this request (ILEN+7)
	QCHK	Check buffer limit
	\$ALC	Request SAM for buffer.
L.06(\$IORQ)		Store request parameter into SAM buffer: 2 T(=1), control, code 3 Priority of requestor 4 Block length 5 User buffer length 6 Optional parameter 1 7 Optional parameter 2 Move data from user buffer into SAM buffer
L.13(\$IORQ)	LINK	Link SAM buffer into the EQT entry.
	DFIVE	Assign a DCPC channel if required
	DRVMP	Set up appropriate map for driver. In this case system map would be used and driver's ptn. would be mapped. Transfer request parameters into EQT entry (words 4-10, and 15).

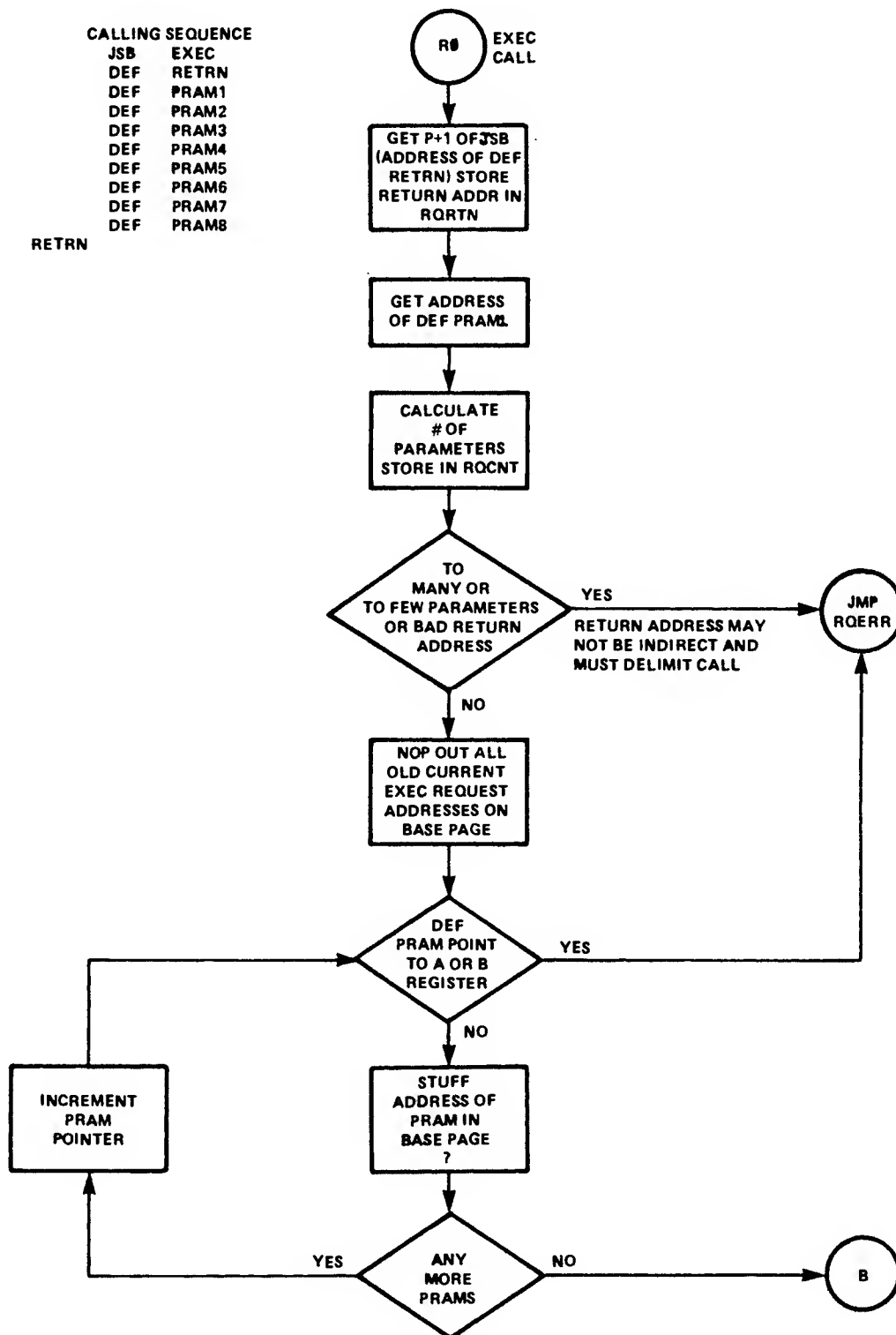
EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
	Ixnn	Enter driver initiator section under system or user map.
	DRIVR	Set EQT availability to busy (2).
\$XEQ		Continue execution of user program.
CONTINUATION INTERRUPTS		
COMPLETION INTERRUPT		
\$CIC(RTIOC)		Save machine state, etc.
	Cxnn	Detect end of data transfer
IOCCM or \$CON1		Clear TO and unlink request from EQT entry.
	\$RTN	Return buffer to SAM
	\$CKLC	Check lower buffer limit and schedule any waiting programs.
L.54(IOC0M)		Set EQT entry to available (AV=0).
L.68(IOC0M)	DRIVE	If EQT entry has I/O requests stacked, initiate next request.
IOCX		Assign DCPC channel if available.
\$4EQ		Continue execution of current user program.

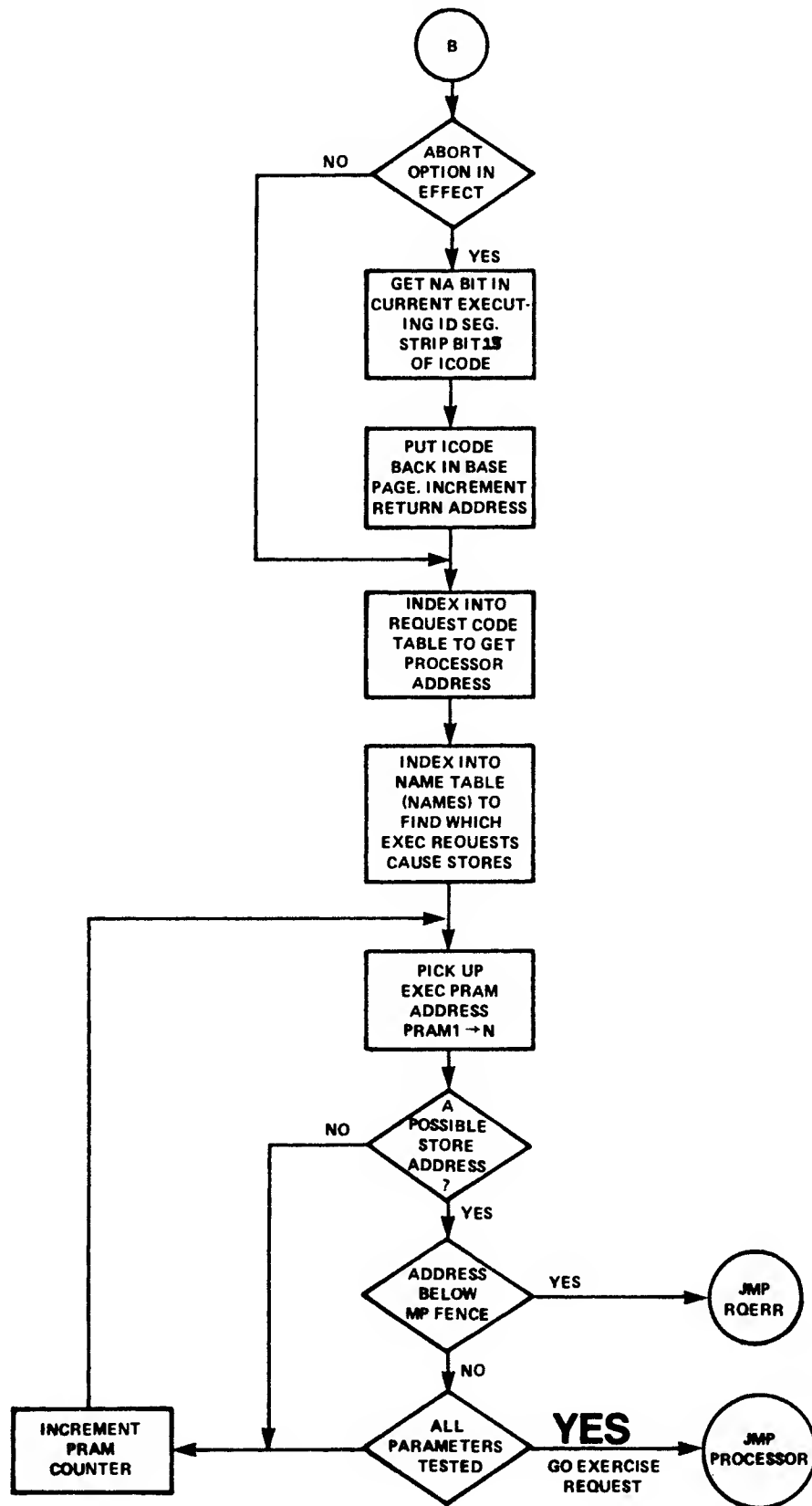




NOTE: DMS VIOLATION IS LEGAL IF A MR PROGRAM CALLS A MR LIBRARY PROGRAM (USEABLE ONLY BY MR PROGRAMS). THE PHYSICAL ADDRESS OF THE LIBRARY WILL BE ABOVE THE MP FENCE IF THE MR PROGRAM IS USING COMMON: HOWEVER, THE PAGES CONTAINING THE LIBRARY ARE WRITE PROTECTED.

CALLING SEQUENCE  
 JSB EXEC  
 DEF RETRN  
 DEF PRAM1  
 DEF PRAM2  
 DEF PRAM3  
 DEF PRAM4  
 DEF PRAM5  
 DEF PRAM6  
 DEF PRAM7  
 DEF PRAM8  
 RETRN







## EXEC CALL PROCESSORS

RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
1			
2, 3			
17, 18	\$IORQ	\$LIST	SUSPEND IF LOCKED
19			NO BUFFER AVAIL.
20			OR SET I/O SUSPEND
		\$ALC	ALLOCATE BUFFER FOR OUTPUT
		\$REIO	MOVE TDB FOR RE-ENTRANT I/O
		\$SYMG	ERROR MESSAGES
		DRIVE	INITIATE I/O
		\$XEQ	RETURN
4, 15	DISC1	\$DREQ	ALLOCATE DISC TRACKS
		\$LIST	SUSPEND IF NOT AVAILABLE
		\$XEQ	RETURN
5, 16	DISC2	\$CREL	RELEASE GLOBAL TRACKS
		\$SDSK	SCHEDULE DISC-SUSP PROGS
		\$XEQ	RETURN
6 (0)	\$MPT1	MPT1B	STANDARD TERMINATION
		\$LIST	SET DORMANT, SCHEDULE FATHER
		\$XEQ	RETURN
(1)		MPID	SAVE RESOURCES TERMINATION
		\$WATR	FIND IF ANY WAITING PROGRAMS
		\$SCD3	IF SO, SCHEDULE 'EM
		\$LIST	SET PROC IN DORMANT LIST, SCHED PO
		\$XEQ	RETURN
(2)		SABRT	SOFT ABORT (SEE 'OF, 0')
		\$TREM	REMOVE FROM TIME LIST
		\$LIST	SET PROC IN DORMANT LIST
		\$XEQ	RETURN
(3)			SEE 'OF, X'
7	\$MPT2	\$LIST	SET SUSPENDED
		\$XEQ	RETURN
8	\$MPT3	\$BRED	SET UP TO READ SEGMENT
		\$XSIO	SYSTEM I/O REQUEST
		\$LIST	SET PROG SUSPENDED
		\$CVEQ	I/O ROUTINES...
		DRIVE	
		\$LIST	SET PROG I/O SUSPEND
		\$XEQ	RETURN

## EXEC CALL PROCESSORS (cont'd)

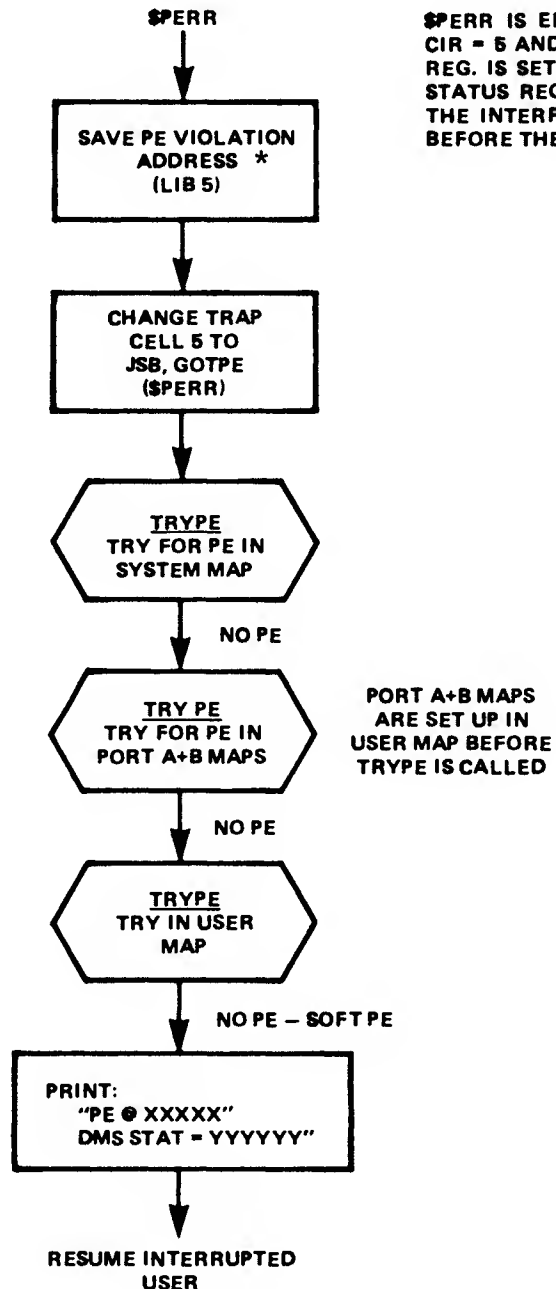
RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
9,23	\$MPT4	IDCKK \$IDNO ALCST \$RTST \$RTN \$LIST \$ALC \$LIST	GET ID SEG NUMBER SET POP POINTER. SAVE PARAMETER STRING RETURN SAM USED FOR STRINGS RETURN MEMORY SCHEDULE MEMORY WAITERS ALLOCATE MEMORY. SCHEDULE 'SCHEULEE' PASS 'BATCH' FLAG,. PLACE SCHEDULER DORMANT
10,24	\$MPT5	IDCKK \$IDNO ALCST \$RTST \$FIN \$LIST \$ALC \$LIST	GET ID SEG. NUMBER SET POP POINTER SAVE PARAMETER STRING. RETURN SAM USED FOR STRINGS RETURN MEMORY SCHEDULE MEMORY WAITERS ALLOCATE MEMORY SCHEDULE 'SCHEDULEE'
11	\$MPT6	\$TIME \$TIMV	ACCESS CURRENT TIME CONVERT & MOVE TO USER
12	\$MPT7	\$TIMR \$LIST \$TADD \$XEQ	DOES THE WORK SET DORMANT IF CURRENT PROG SET IN TIME LIST RETURN
13	\$IORQ	\$CVEC \$XEQ	GET EQ1 ADDRESS RETURN
22	\$MPT8	\$XEQ	DOES THE WORK RETURN
14(1)	\$MPT9	\$STSH \$RTST \$RTN \$LIST	GET (RETRIEVE) STRING GET ADDRESS OF STRING RETURN STRING MEMORY RETURN MEMORY SCHEDULE MEM.WAIT PROGS.

RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
-----	-----	-----	-----
(2)		MPT9W ALCST \$RTST \$RTN \$LIST	PUT(WRITE)STRING TO FATHER SAVE PARAMETER STRING. RETURN STRING MEMORY RETURN MEMORY SCHEDULE MEM.WAIT PROGS.
25	\$PTST	\$IDNO	GET ID SEG. OF OCCUPANT
26	PISIZ		DOES THE WORK



## PARITY ERRORS

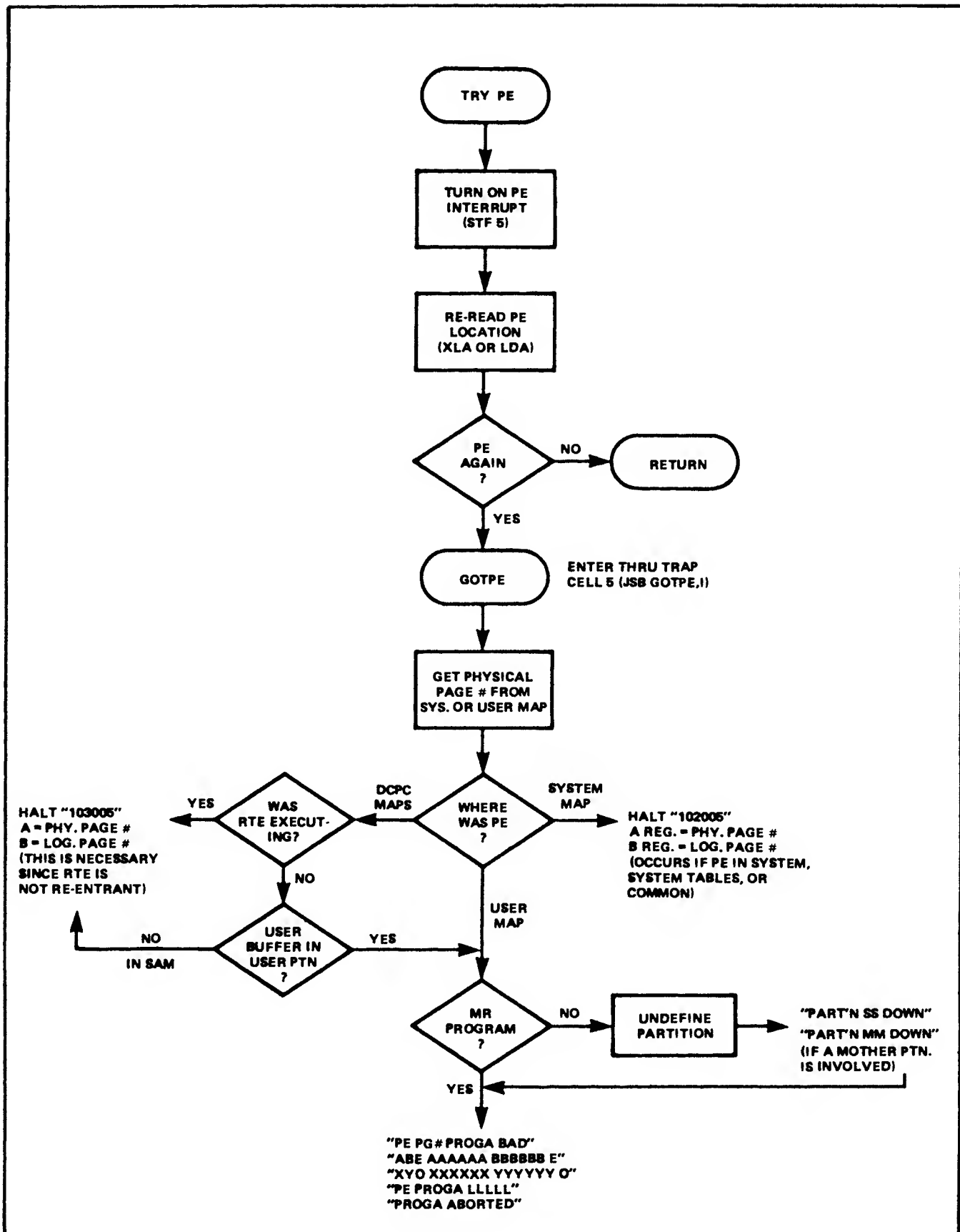




**\$PERR IS ENTERED FROM \$CIC IF CIR = 5 AND BIT 15 OF VIOLATION REG. IS SET. \$CIC SAVES THE DMS STATUS REG. AND SETS A FLAG IF THE INTERRUPT SYSTEM WAS OFF BEFORE THE PE INTERRUPT.**

**PORT A+B MAPS ARE SET UP IN USER MAP BEFORE TRYPE IS CALLED**

**\* ON A PE THE VIOLATION REGISTER CONTAINS A COPY OF THE M REGISTER.**





## RESOURCE NUMBERS



# RESOURCE MANAGEMENT

## ( RESOURCE NUMBERING)

ALLOWS COOPERATING PROGRAMS A METHOD OF SHARING RESOURCES

CALL RNRQ (ICODE, IRN, ISTAT)

15	14	5	4	3	2	1	0
WAIT OPTION		ALLOCATE OPTION			LOCK OPTION		
NO W A I T	NO A B O R T	C L E A R	G L O B A L	L O C A L	C L E A R	G L O B A L	L O C A L

IRN = RESOURCE NUMBER. RETURNED ON ALLOCATE;  
REQUIRED OTHERWISE.

ISTAT =  
(RETURNED)

- 0 NORMAL DEALLOCATE
- 1 RN IS CLEAR (UNLOCKED)
- 2 RN IS LOCKED LOCALLY TO CALLER
- 3 RN IS LOCKED GLOBALLY
- 4 NO RN AVAILABLE NOW
- 5 —
- 6 RN IS LOCKED LOCALLY TO ANOTHER PROGRAM
- 7 RN WAS LOCKED GLOBALLY WHEN REQUEST WAS MADE

NOTE: STATUS 4, 6, AND 7 ARE RETURNED ONLY IF  
"NO WAIT" BIT IS SET.

# RESOURCE NUMBER TABLE

\$RNTB		# of RNs			1
RN 1		OWNER1		LOCKER1	2
RN 2		OWNER2		LOCKER2	3
		.		.	
		.		.	
		.		.	
		.		.	
RN N		OWNERn		LOCKERn	N+1

## OWNER/LOCKER ENTRY:

- ID segment number in keyword block of owner/locker program
- 377B if globally owned/locked
- 0 if not owned or locked

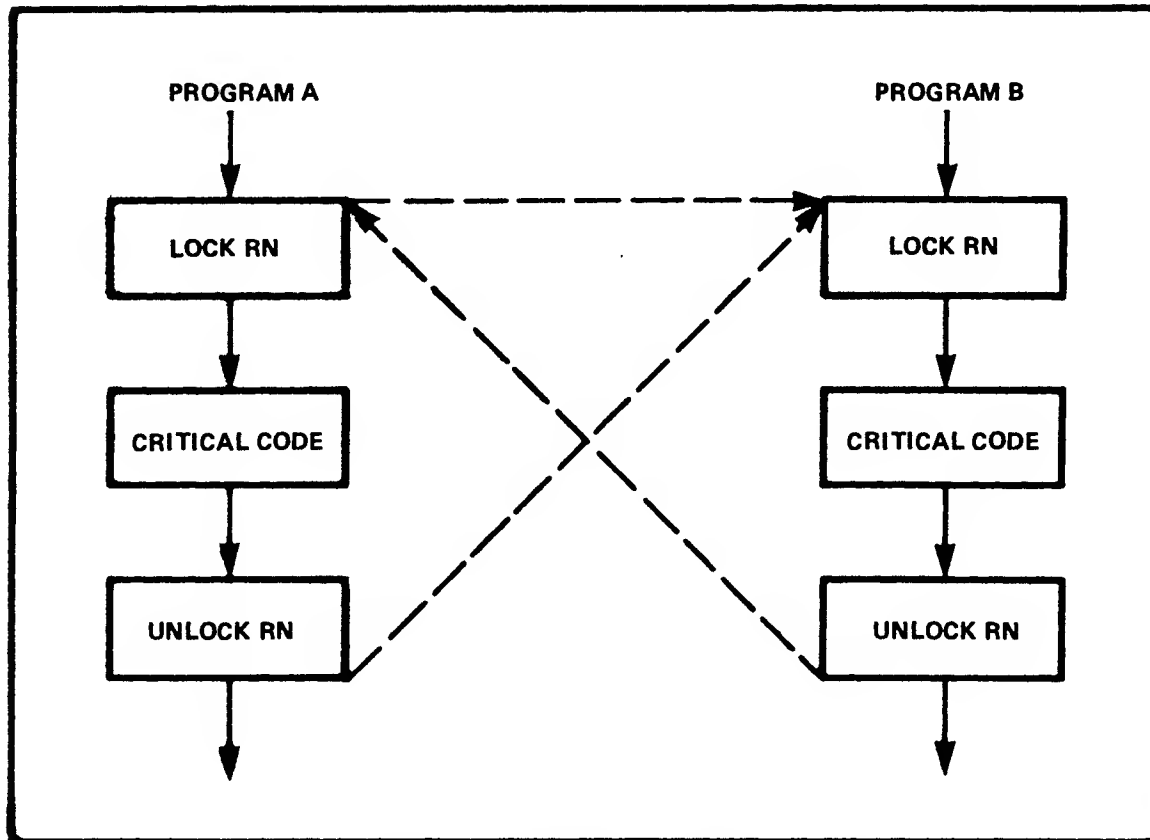
# RESOURCE NUMBER FORMAT

15		8	7		0
	OWNER ENTRY FROM				RESOURCE NUMBER
	RN TABLE				(1-N)

## RN MANAGEMENT

- RNRQ manages RN's
- "RN" bit in ID segment word 21 is set when a program is a RN owner or locker.
- RNRQ is a type 6 utility subroutine
- Programs attempting to lock a locked RN are put into general wait (3)
- When a program aborts or terminates, the system (\$TRRN):
  - Releases the program's local RN locks
  - Deallocates its local RN's
  - Reschedules waiting programs

# RN APPLICATIONS



**RESOURCE NUMBER (RN) LOCKING ALLOWS TWO OR MORE COOPERATING PROGRAMS TO ACCESS SENSITIVE AREAS OF THEIR CODE ON A ONE-AT-A-TIME-ONLY BASIS**

**CRITICAL CODE MIGHT REFERENCE SHARED:**

- 1. COMMON AREAS**
- 2. DATA BASE**
- 3. PERIPHERAL DEVICES**
- 4. DISC FILES**

LOGICAL UNIT LOCK





# LOGICAL UNIT LOCK

ALLOWS A PROGRAM TO EXCLUSIVELY DOMINATE (LOCK) A GROUP OF I/O DEVICES

DIMENSION LUARY (number of LU's to be locked)  
 CALL LURQ (IOPTN, LUARY, NOLU)

000000B = UNLOCK SPECIFIED LU'S	↑	↑	↑
100000B = UNLOCK ALL LU's	↑	↑	↑
000001B = LOCK WITH WAIT			
100001B = LOCK WITHOUT WAIT			

↑  
4 = NO ABORT ON ERRORS

ARRAY OF LU'S  
TO BE LOCKED

NUMBER OF LU'S  
TO BE LOCKED

ON RETURN:

"A" REGISTER = 0 LU LOCK  
SUCCESSFUL  
 =>1 LU LOCK NOT SUCCESSFUL  
 = -1 NO RESOURCE  
 NUMBER AVAILABLE  
 = 1 ONE OR MORE OF THE  
 LU'S ALREADY LOCKED

THIS CALL USES RESOURCE NUMBERS

## LU LOCK MANAGEMENT

- LURQ (a type 6 utility subroutine) manages LU locks
- LURQ flow is:
  - Allocate a local RN for the calling program
  - Locally lock the RN to the caller
  - Enter RN into DRT entry for each LU
- Programs attempting to use or lock a locked LU are put into general wait (3)
- A maximum of 31 programs may simultaneously lock LUs
- LU locks are removed by \$TRRN when the program:
  - Terminates
  - Terminates serially reusable
  - Aborts

# LU LOCK EXAMPLE

PROGRAM XYZ

```

      .
      .
      .
LUARY(1)=5
LUARY(2)=7
CALL LURQ(1,LUARY,2)
      .
      .
      .

```

SYSTEM TABLES WOULD BE:

DRT PART 1			
		.	
		.	
		.	
		.	
5		2	
6	SUB	RN	EQT
	CHAN		
7		2	
		.	
		.	
		.	
		.	

RN TABLE		
	N	
		1
	15*	15
		2
	.	
	.	
	.	
	.	
	.	
	.	
	.	
	.	
	.	N+1

\*Assumes that Program XYZ's keyword block index to its ID segment is 15.



## PROGRAM STATES



## ***PROGRAM STATES***

•	Dormant	0
•	Scheduled	1
•	Executing	1
•	Suspended:	
	I/O suspend	2
	unavailable memory	4
	disc allocation	5
	operator suspend	6
•	General Wait	3

The General Wait state is implimented to indicate a waiting state in which a program is swappable.  
Reasons for wait include waiting for:

- a. buffered I/O
- b. class I/O "Get"
- c. Resource Number lock/availability
- d. I/O class availability
- e. program scheduled with wait
- f. scheduling in queue
- g. downed I/O device
- h. LU lock
- i. buffer limit exceeded

Dormant substates include:

- a. terminate sarring resources
- b. terminate serial re-useable
- c. truly not active

**RTE STATE CHART**  
**(handout)**



## PROGRAM STATE LISTS

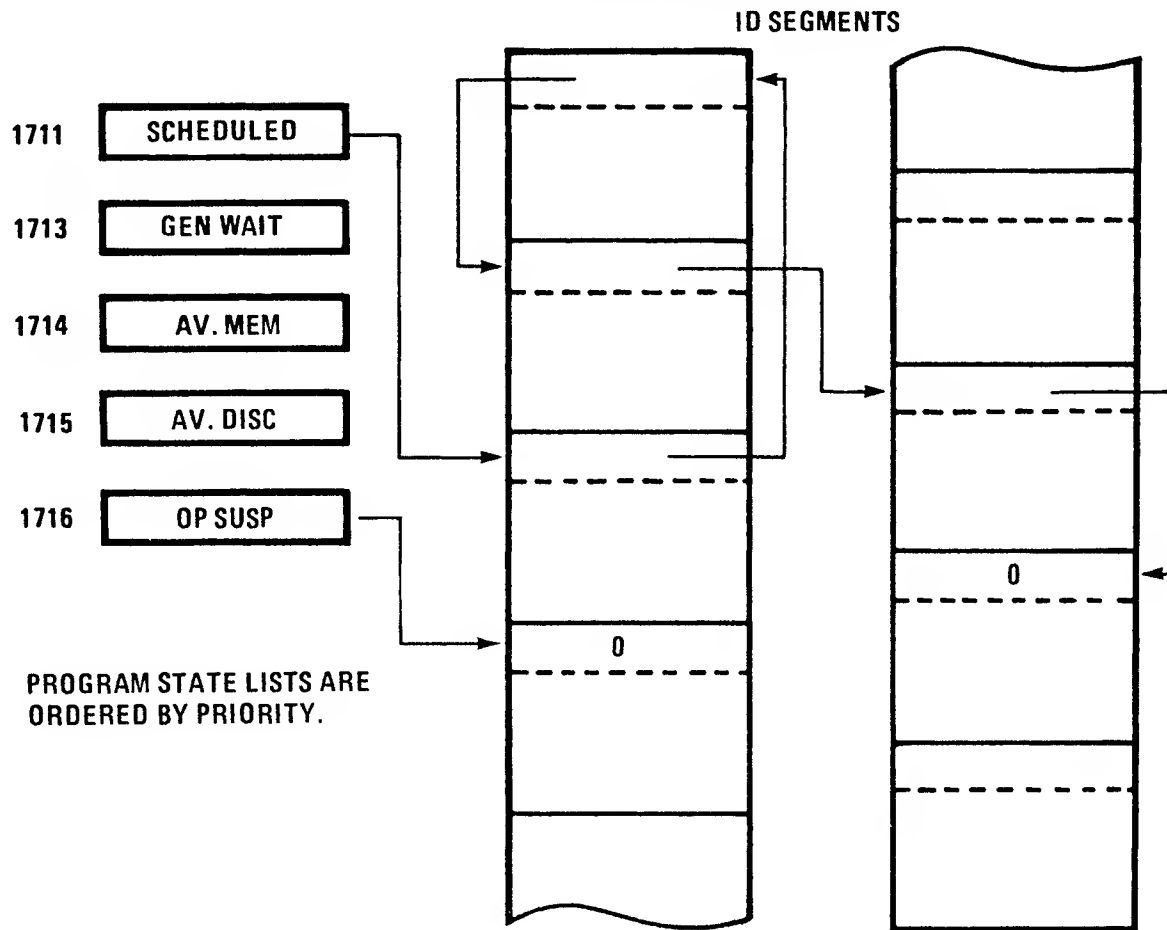
RTE moves programs from state to state by linking and unlinking the programs ID segment between the appropriate state lists.

Programs in each state are grouped as follows:

STATE -----	LISTHEAD -----
- Dormant (0)	none
- Scheduled (1)	SKEDD(1711)*
- I/O Suspend (2)	each EQT entry
- General wait (3)	SUSP2(1713)
- Memory (4)	SUSP3(1714)
- Disc Allocation (5)	SUSP4(1715)
- Operator Suspend (6)	SUSP5(1716)

\* Octal base page addresses

# LIST LINKING



ID link word is word #1.

# STATE REPRESENTATION IN PROGRAM'S ID SEGMENT

ID SEGMENT  
WORD(BITS)

CONTENTS

-----

1 Linkage(or 0)

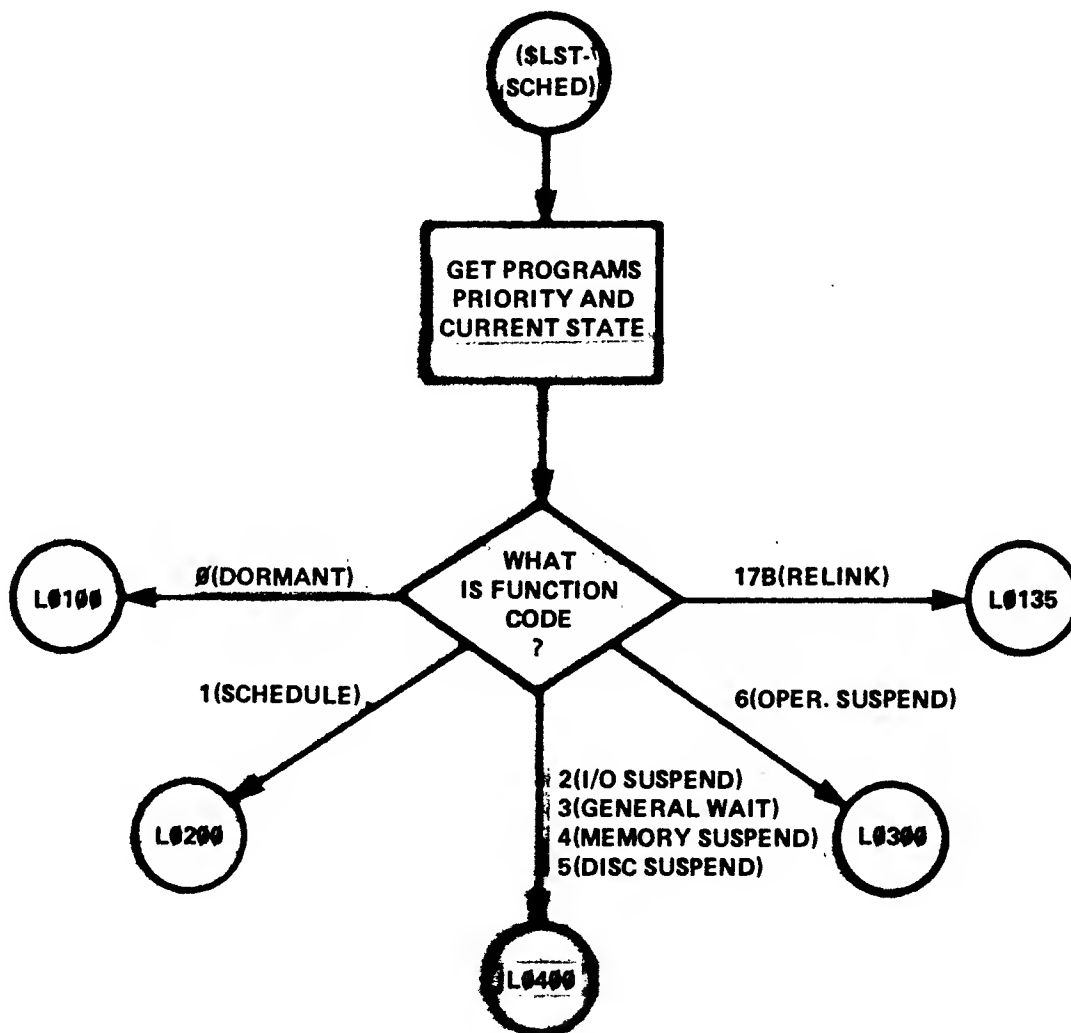
2 I/O Suspend(2): EXEC call request parameters in  
ID2 thru ID6.

General wait(3):

- a. ID segment address of program waiting to schedule
- b. ID segment address of son waiting to complete
- c. Address of RN table (\$RNTB) if waiting on RN allocation.
- d. Address of RN if waiting on a RN lock.  
\$RNTB < ID(2) < \$RNTB + (\$RNTB)
- e. "4" if waiting on a downed device. Also ID(3) will be the LU# of the device.
- f. Address of class table (\$CLAS) if waiting on class allocation.
- g. Address of class number if waiting on a class "GET". \$CLAS < ID(2) < \$CLAS + (\$CLAS).
- h. Address of RN if waiting on an LU lock. DFT will also contain the RN.
- i. EQT entry address on which the buffer limit has been exceeded.

STATE REPRESENTATION INn  
PROGRAM'S ID SEGMENT (cont'd)

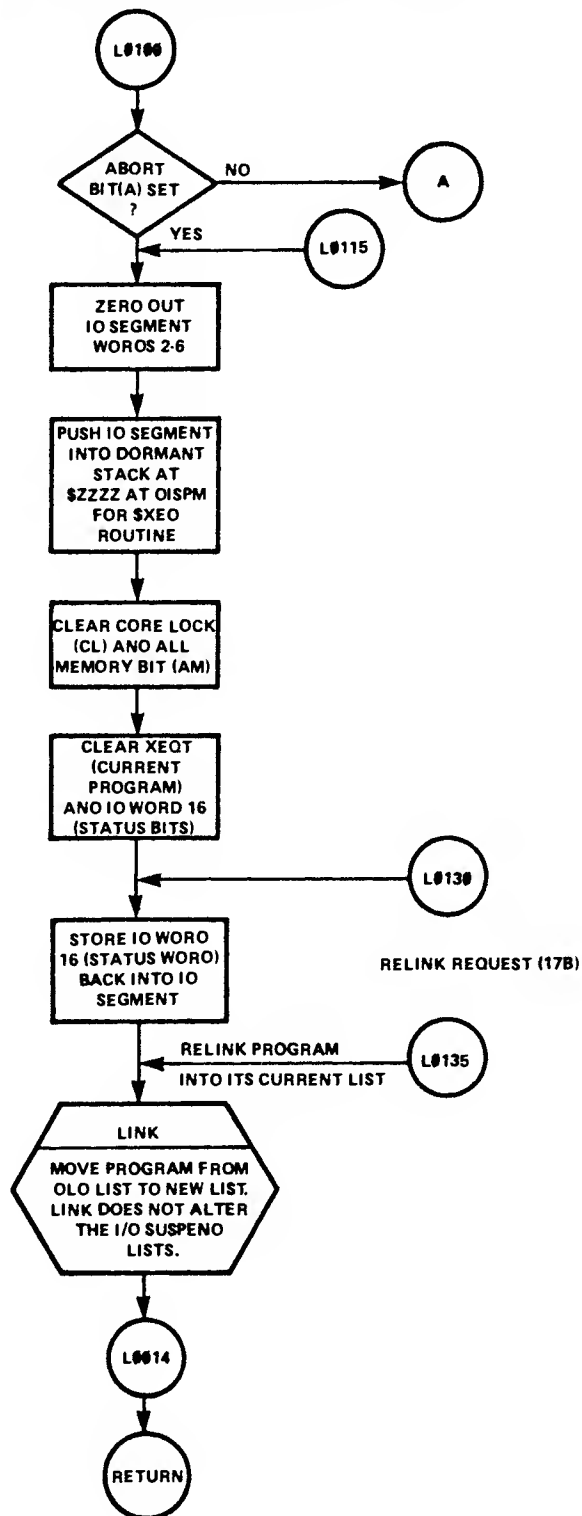
WORD (BITS)	CONTENTS
2 cont'd	Memory (4): Number of SAM words requested
16(3-0)	Program state (0-6)
16(6)	Dormant (D) bit: set program dormant on next schedule attempt.
16(7)	Save resources (R) bit: program wants to save it's resources when it goes dormant. (R bit is cleared when set dormant.)
16(9)	Operator suspend (O) bit: suspend program as soon as feasible.
16(11)	Abort (A) bit: abort program and set dormant as soon as feasible.
16(12)	Wait (W) bit: this program is waiting to schedule another program.

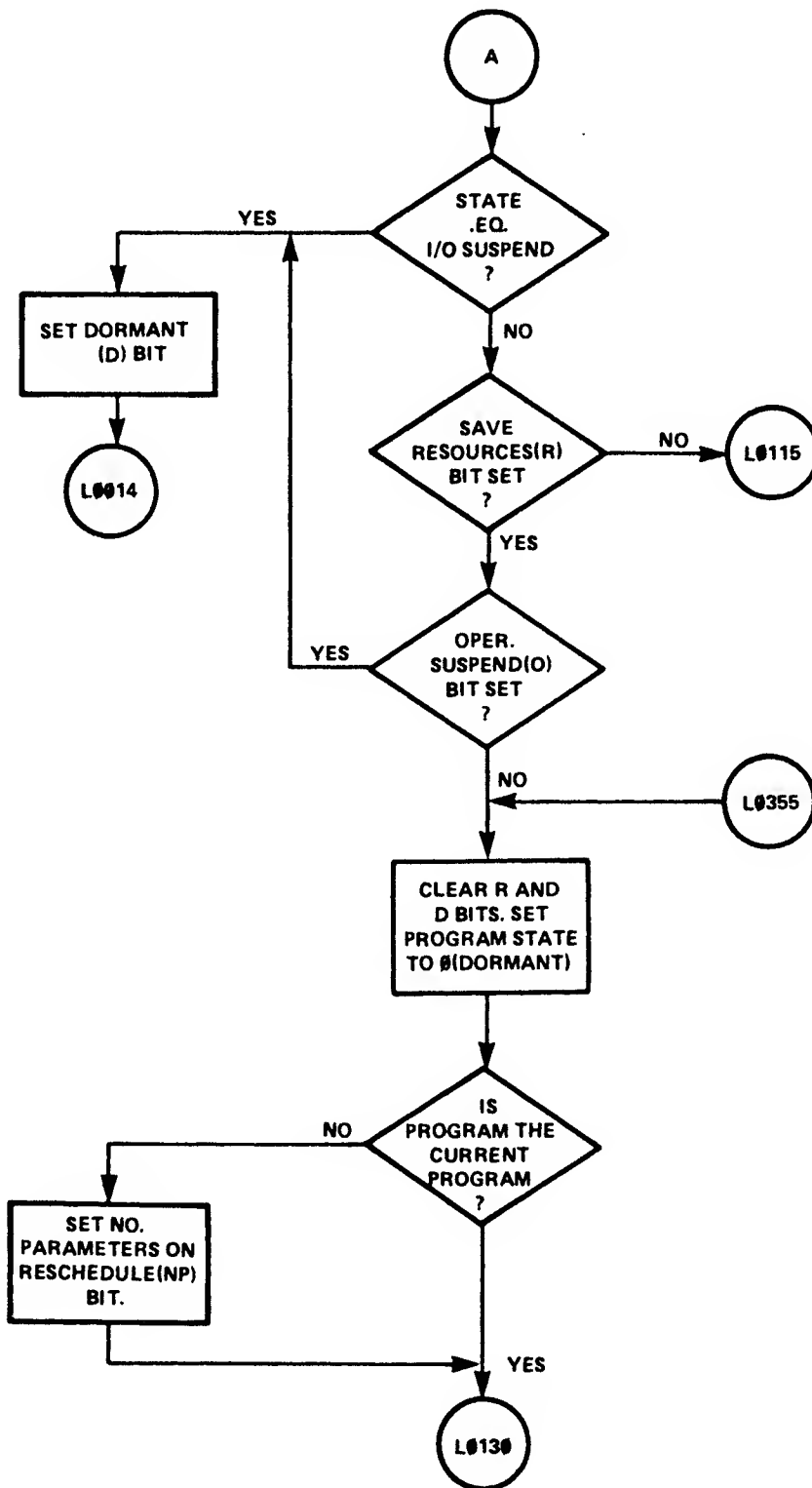


#### CALLING SEQUENCE

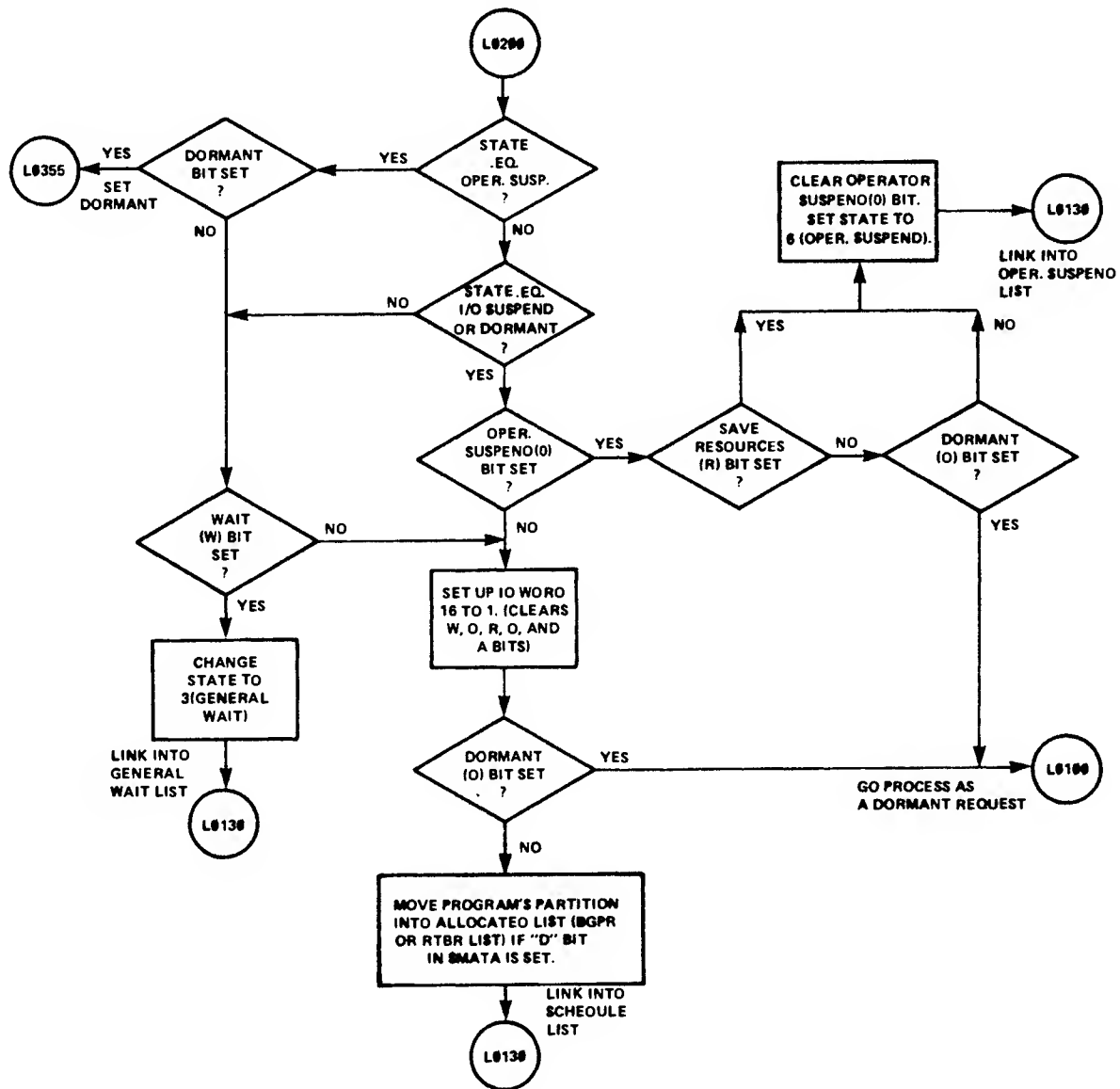
JSB	\$LIST
OCT	(ADDRESS CODE) (FUNCTION CODE: 0-6 AND 17B)
DEF	ID SEGMENT ADDRESS

# DORMANT REQUEST (0)



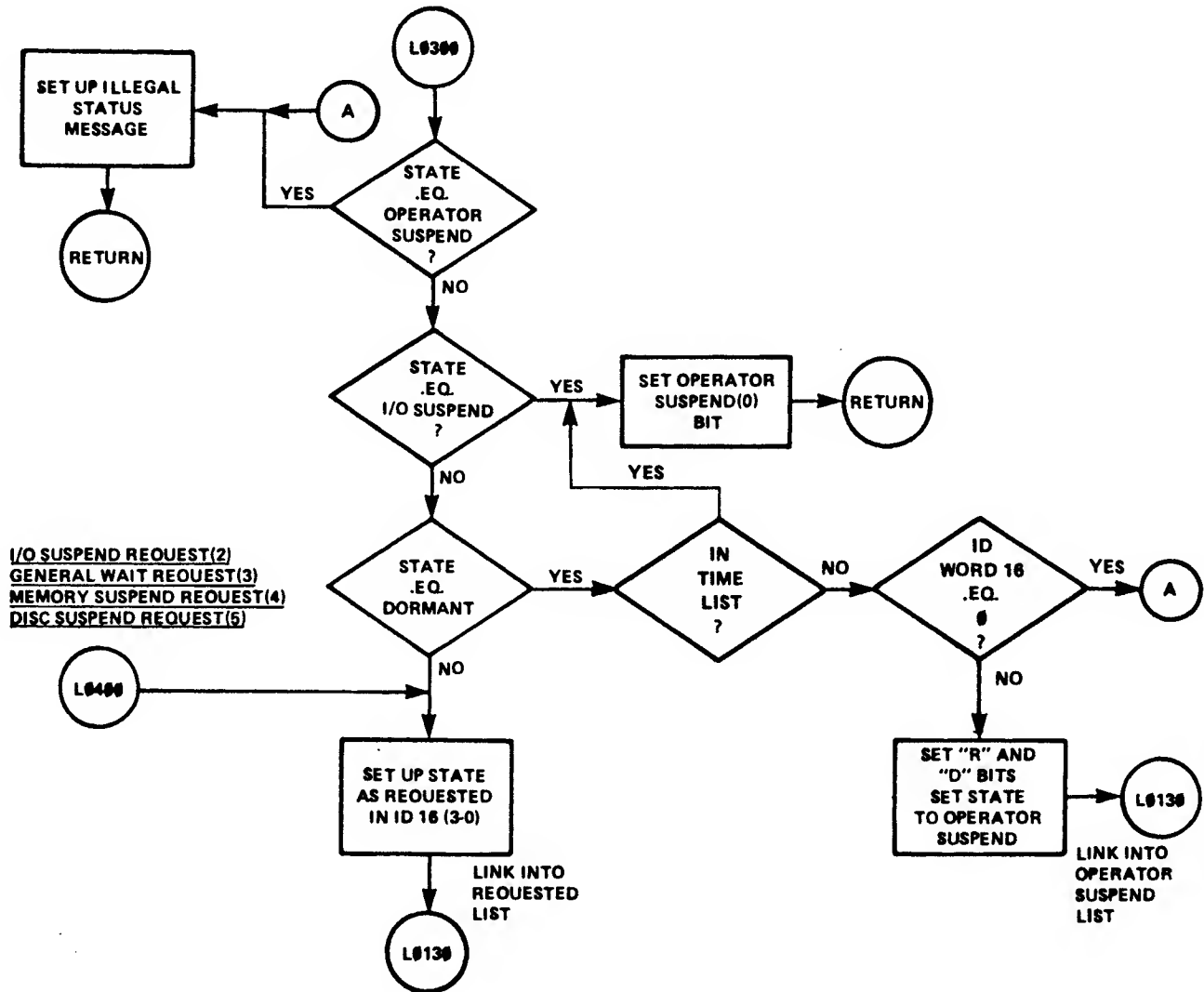


## SCHEDULE REQUEST (1)





# OPERATOR SUSPEND REQUEST ⑥



# WHZAT

0032\*THE FOLLOWING IS A SAMPLE OUTPUT OF THIS PROGRAM:

```

0033*          ON,WHZAT,LU
0034*
0035* 09:51:50:710
0036* *****
0037* PT SZ PRGRM,T ,PRIOR*DRMT*SCHD*I/O *WAIT*MEMY*DISC*OPER * NEXT TIME *
0038* *****
0039* 0 ** MEM *1 *00000 ***** 1
0040* 0 ** RSPNS*1 *00010 ***** 3,CL 032
0041* 3 5 PROGA*3 *00097 ***** 6
0042* 4 5 PROGB*3 *00097B ***** 3,LULK 40,LKPRG=PROGA
0043* 5 17 PROGC*3E*00097 ***** 3,RN 031,LKPRG=PROGD
0044* 3A27 PROGD*4 *00097 ***** 3,RESOURCE
0045* 5 7 PROGE*3 *00097 ***** 3,CLASS #
0046* 2 4 QUICK*3 *00099 0 *****00:00:00:000
0047* 6 7 FMGR *3 *00090 ***** 3,EDITR'S QUEUE
0048* 3 7 EDITR*3 *00050 ***** 5
0049* 6 15 ASMB *3 *00099 ***** 3,LU,EQ DN 6, 5(0[00000000])
0050* 4A 6 TIMEL*4 *00090 ***** 3,LU,EQ DN 6, 5(0[00000000])
0051* 4A 6 TIMEL*4 *00090 ***** 3,LU,EQ DN *****00:00:00:000
0052* 7 7 FMG07*3 *00050 ***** 3,BL,EQT 7
0053* 2 3 WHZAT*4 *00001 ***** 1
0054* 0 ** RENSB*1 *00060 ***** 4
0055* 3 6 PROGF*4 *00096 ***** 3,RN 031,LKPRG=GLOBL
0056* 6 7 ED26 *3 *00050 ***** 2, 16(2[00000010])
0057* *****
0058* DOWN LU'S, 6, 14
0059* *****
0060* DOWN EQT'S, 5, 6
0061* *****
0062* 09:51:50:710
0063*

```

Reason for I/O Suspend: EQT entry number (STATUS field of EQT entry word 5[binary content of STATUS])

Reason for wait:

BL,EQT eqt	buffer limit exceeded on the controller in EQT entry eqt
CL ccc	waiting for class number ccc to complete GET
CLASS #	waiting for a class number
LU/EQ DN	A device or controller is down. Look at DOWN LU's or DOWN EQT'S list at bottom of report
LULK lu,LKPRG= prog name	Logical unit with number lu is locked to named program
program name	waiting for named program to complete
program'S QUEUE	waiting to schedule named program which is busy
RESOURCE	waiting for resource number
RN nnn,LKPRG= prog name	Resource number nnn is locked to named program

PT SZ COLUMN HEADING (PARTITION NUMBER AND PARTITION SIZE)

0 \*\* MEMORY RESIDENT PROGRAM

5 8 PARTITION #5 IS USED AND 8 PAGES IN USE

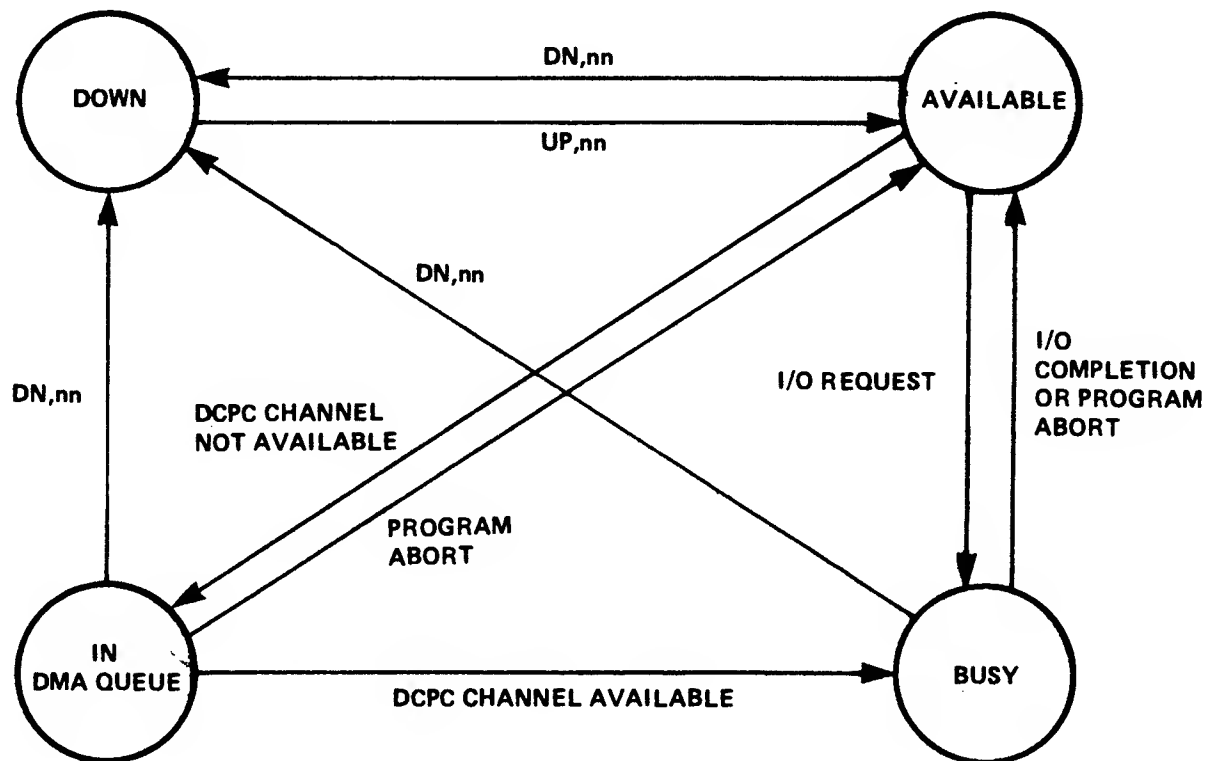
11 SCHEDULED PROGRAM IS NOT YET IN PARTITION

'A' AFTER THE PARTITION # MEANS THE PROGRAM WAS ASSIGNED

'E' AFTER THE PROGRAM'S TYPE MEANS IT IS AN EMA PROGRAM

'B' AFTER THE PROGRAM'S PRIORITY MEANS RUNNING UNDER BATCH

# EQT ENTRY STATE DIAGRAM



EQT entry is kept in EQT entry word 5 in the AV field where:

- 0 = Available
- 1 = Down (only with a "DN" command)
- 2 = Busy
- 3 = Waiting for DCPC channel

## HANDLING OF "DISPLACED" I/O BUFFERS DUE TO A DOWNED DEVICE

Subchannels of a device are downed because of:

- Device-time out
- Device not ready
- Parity error
- "DN,,LU" command

I/O requests queued to the subchannel's EQT entry are removed and the follow action is taken depending on the I/O request type:

### USER NORMAL OPERATION

-----

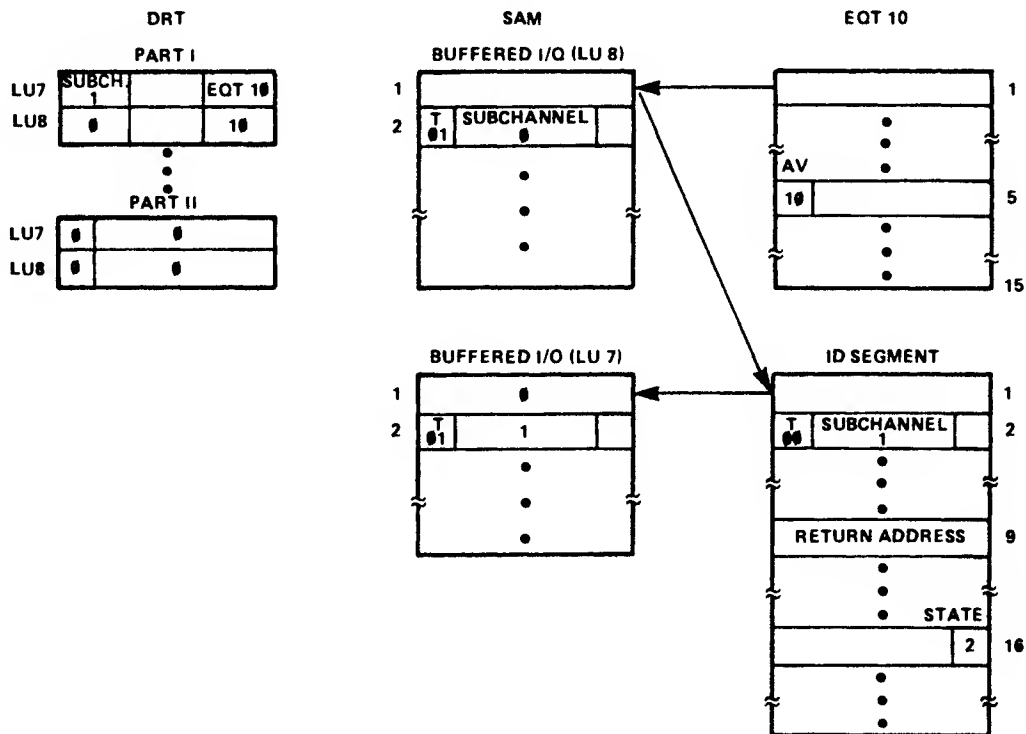
The program's point of suspension (ID word 9) is backed up from the return address to the address of the "JSB EXEC" (saved in ID word 10). A "4" is stored into ID word 2, the program's state is changed to general wait (3), and its ID segment is linked into the general wait list.

### USER AUTOMATIC OUTPUT BUFFERING

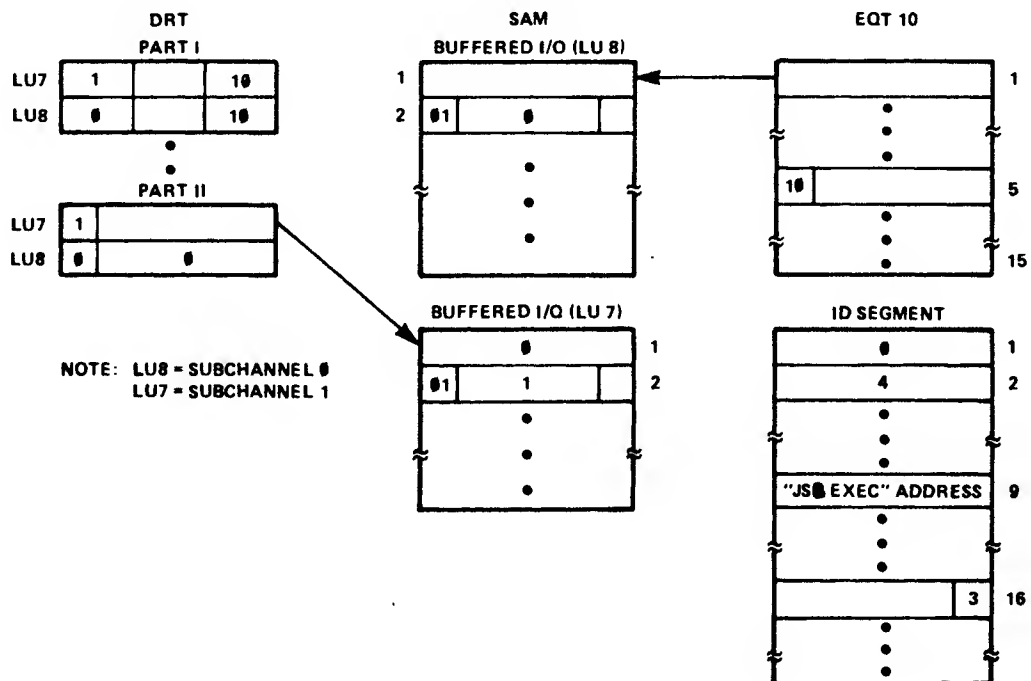
-----

The buffer is unlinked from the EQT entry and relinked into the second half of the DRT table on the associated LU. Bit 15 of the DRT is set to indicate the LU is down. CLASS I/O and SYSTEM I/O request are handled in the same manner.

# BUFFER HANDLING EXAMPLE



\*DN,,7





TBG TIME TICK





## TBG INTERRUPT

### ENVIRONMENT BEFORE INTERRUPT:

1. TBG (Time Base Generator) on select code 11
2. System time is ten milliseconds before 8:00 a.m.
3. PROGA is in the time list and scheduled to run at 8:00 a.m.
4. PROGB is also in the time list and scheduled to run at 9:00 a.m.
5. PROGC is currently executing.

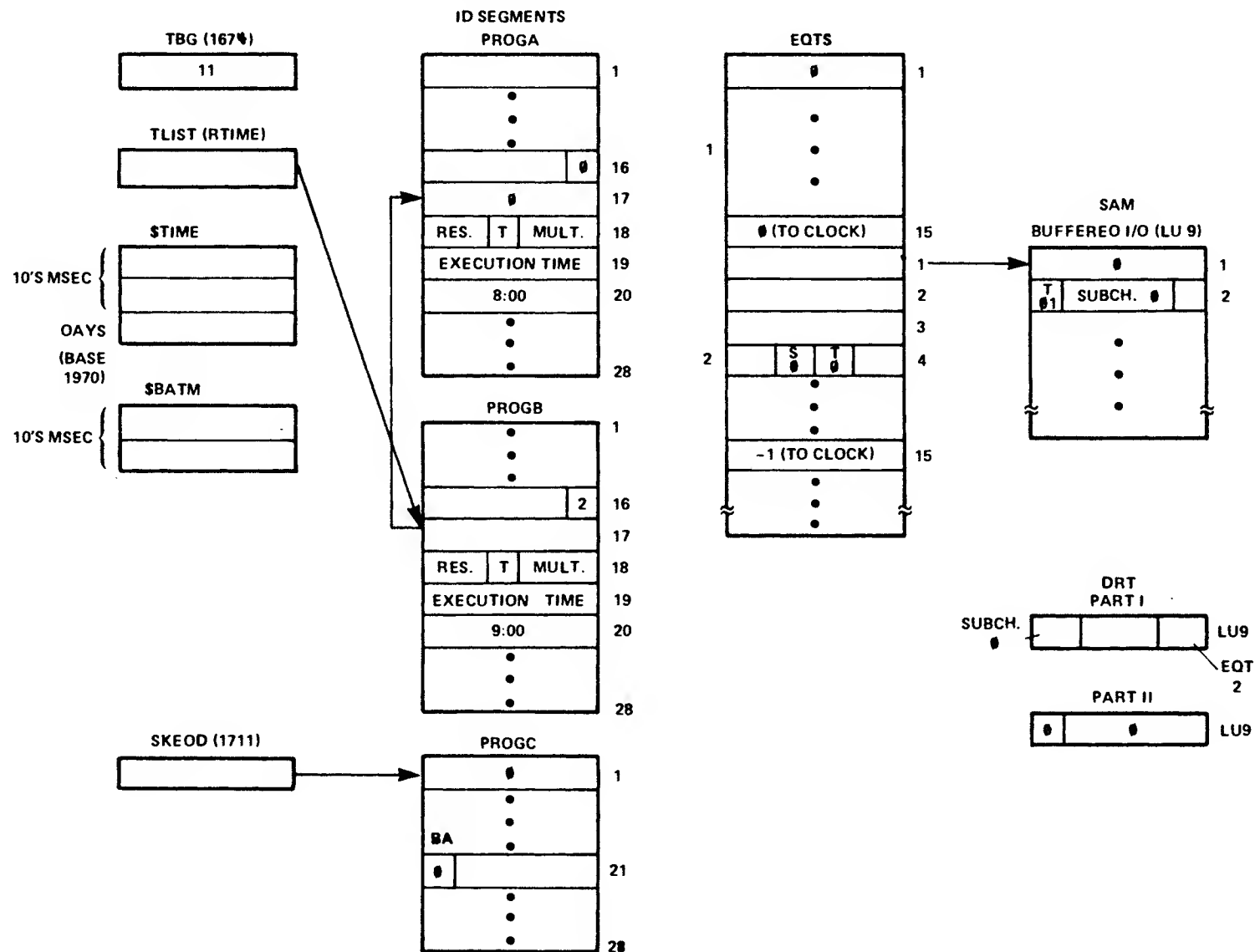
# TIME TICK PROCESSING

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
TBG interrupt		TBG generator interrupts every 10 milliseconds (msec)
TRAP CELL 11(JSB \$CIC,I)		Interrupt causes the instruction in trap cell 11 to be executed
\$CIC(RFIOC)		Saves the machine state and turns off the interrupt system. Compare select code of interrupt (CIR=11) to base page (BP) TBG word (1674B).
\$CLCK(RTIME)		Step the system time kept in RTIME (\$TIME) in a double word integer in 10's of msec. The first word is stepped (\$TIME) and if it goes to zero, the second word (\$TIME+1) is stepped. If the second word goes to zero it is midnight and the double integer is reset for the next day and the day word (\$TIME+2) is stepped. Days are kept in one word referenced to the base year (1970).

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
CL010 (\$CLCK)		Compare system time to next execution time (ID words 19 and 20) of each program on time list (threaded thru ID word 17).
	TMSCH(\$CLCK)	If times are equal, the program should be scheduled.
	\$LIST	If the program is in the dormant state (ID word 16), schedule it.
	\$TREM	If the program's MULTIPLE value (ID word 18) is zero, remove the program from the time list.
	TUDAT	If the MULTILE value is not zero, use it and the RESOLUTION value (ID word 18) to calculate the program next execution time.
TOBAT (\$CLCK)		Step the batch time (\$BATM) if the currently executing program is a batch program (ID word 21-BA bit) and not SMP or D.RTR.
	\$ERMG	If batch time is zero, abort current batch program.

EXECUTION -----	SUBROUTINE CALLS -----	NOTES -----
IOTOP(\$CLCK)		Step the time out clock (EQT entry word 15) in each EQT entry which has a time out in progress. (EQT word 15 not zero) If none of the time out clocks go to
\$XEQ(DISPM)		zero then dispatch the next program
\$DEVT(RTIOC)		else the EQT entry has timed out. Set the time out bit (T) in EQT entry word 4.
CIC.6(\$CIC)		If the driver will handle the time outs (EQT entry word 4-S bit); enter the driver's continuation/completion section (Cxnn).
IOCOM-\$CON1(RTIOC)		If the driver is not to handle the time out, a driver completion return is simulated by transferring to the I/O completion routines with an error code of 4 (time out). A time out message will be output and the associated LU's will be set down.
XEQ(DISPM)		Dispatch the next program.

# TIME TICK TABLES/LISTS (INITIAL CONDITIONS)

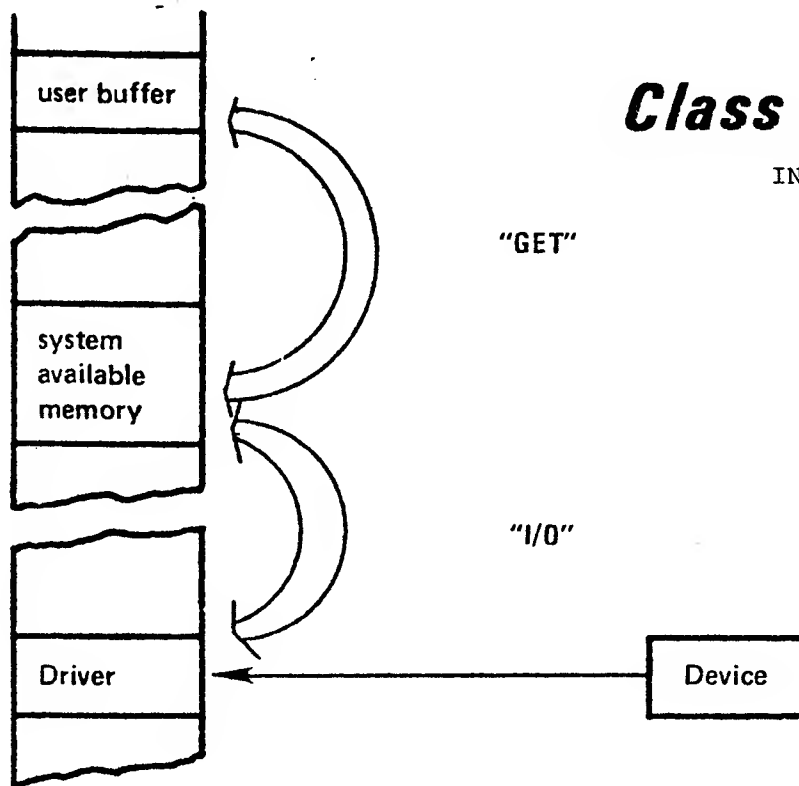




CLASS I/O







## ***Class I/O***

INCLUDES:

1. I/O WITHOUT WAIT
2. PROGRAM-TO-PROGRAM I/O

In Class I/O, the user area is buffered in System Available Memory in a block of memory identified by a "Class Number". The user is thus swappable. The data is retrieved with a "Class Get" call to the appropriate class number.

NOTE NOTE NOTE

All Class I/O is double-call I/O: One call to initiate the operation  
and

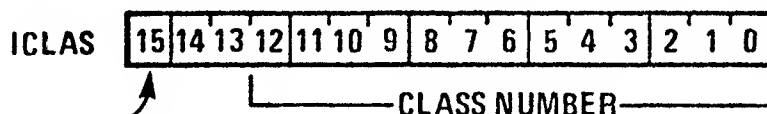
One call to complete the operation

# CLASS I/O-READ/WRITE

TRANSFERS DATA TO OR FROM AN I/O DEVICE. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

DIMENSION IBUFR (size)  
 ICODE = (17, READ; 18, WRITE; 20, WRITE THEN READ)  
 ICLAS = 0, ALLOCATE A CLASS NUMBER;  
 1-255, A CLASS NUMBER TO USE

CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IPRM1, IPRM2, ICLAS)  
 SAME AS FOR ICODE = 1 OR 2      USER INFORMATION  
 (IBUFR IS A DUMMY      PASSED TO GET CALL  
 VARIABLE FOR ICODE = 17)



NO WAIT BIT	}	= 0, PROGRAM IS PUT IN GENERAL WAIT LIST (STATE 3)	}	ON RETURN FROM CALL
		IF MEMORY OR CLASS NUMBER NOT AVAILABLE.		
		= 1, "A" REGISTER = -1, NO CLASS NUMBER AVAILABLE		
		"A" REGISTER = -2, NO MEMORY AVAILABLE		
		"A" REGISTER = 0, SUCCESSFUL CALL		

\* WRITE/READ IS USED WITH LU-Ø FOR PROG. TO PROG. COMMUNICATION

# CLASS I/O - I/O CONTROL

TO PERFORM VARIOUS I/O CONTROL OPERATIONS SUCH AS BACKSPACE, WRITE END-OF-FILE, REWIND, ETC.. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

CALL EXEC (19, ICNWD, IPRAM, ICLAS)

SAME AS FOR  
STANDARD I/O  
CONTROL CALL  
(REQUEST  
CODE = 3)

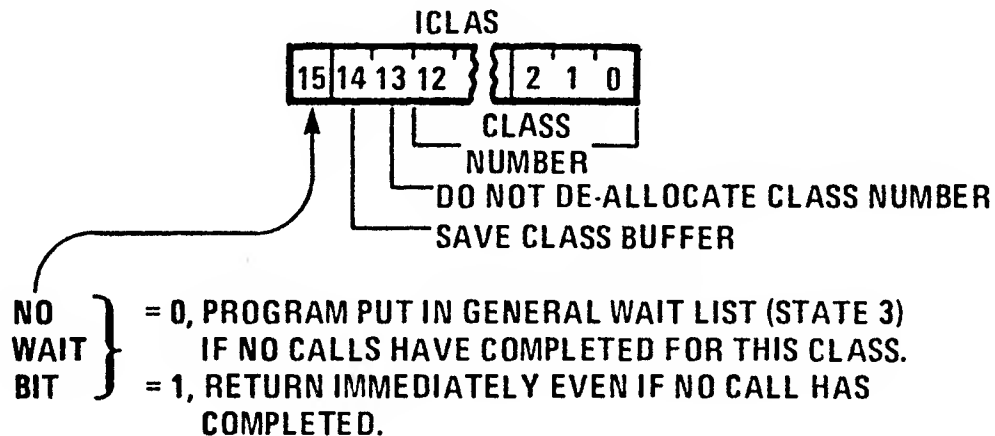
SAME AS FOR  
CLASS I/O -  
READ/WRITE



# CLASS I/O-GET

COMPLETES THE DATA TRANSFER BETWEEN THE SYSTEM AND USER PROGRAM THAT WAS PREVIOUSLY INITIATED BY A CLASS REQUEST.

CALL EXEC (21, ICLAS, IBUFR, IBUFL, IRTN1, IRTN2, IRTN3)



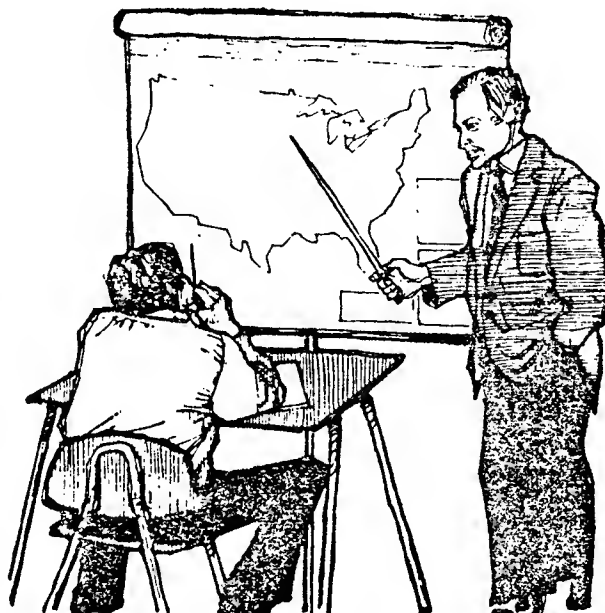
A AND B REGISTERS  
ON RETURN:

## SUCCESSFUL GET

A REG, BIT 15 = 0  
A REG = STATUS  
B REG = TRANSMISSION LOG

## UNSUCCESSFUL GET

A REG, BIT 15 = 1  
A REG = NEGATIVE OF  
(NUMBER + 1) OF  
REQUESTS NOT COM-  
PLETED FOR THIS CLASS



# CLASS I/O-GET (cont.)

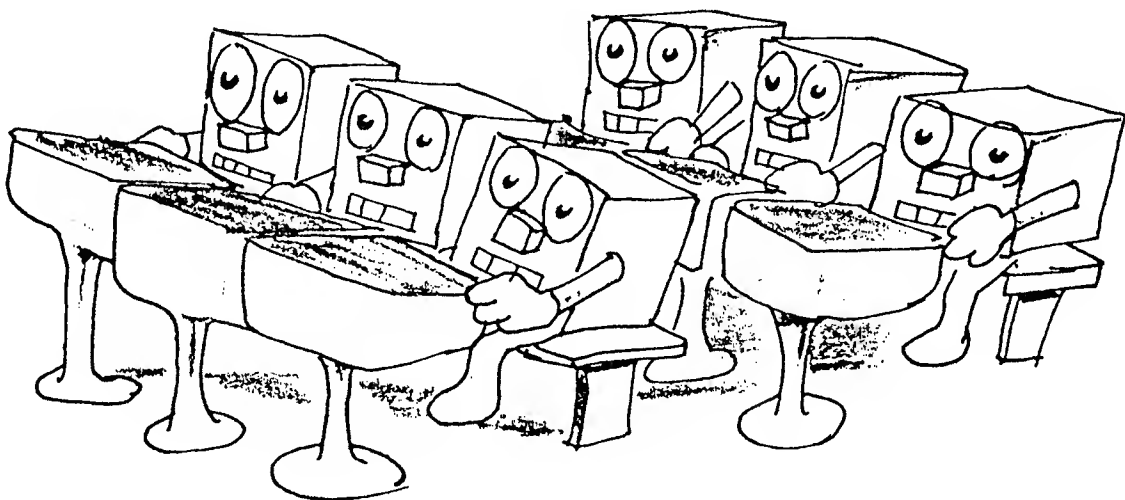
IBUFR DATA IS RETURNED HERE FROM CLASS READ (17)  
OR WRITE/READ (20) CALLS. IT IS A DUMMY VARIABLE  
FOR CLASS WRITE (18) AND CONTROL (19) CALLS.

IBUFL DATA BUFFER LENGTH; WORDS (+), CHARACTERS (-)

IRTN1, USER INFORMATION PASSED FROM CLASS READ, WRITE  
IRTN2 OR WRITE/READ CALLS

IRTN3 REQUEST CODE RECEIVED BY DRIVER RETURNED HERE

<u>ORIGINAL REQUEST CODE</u>	<u>VALUE RETURNED IN IRTN3</u>
17/20 (READ, WRITE/READ)	1
18 (WRITE)	2
19 CONTROL	3



# CLASS TABLE

-----

THE CLASS TABLE ENTRY CAN BE IN ONE OF FOUR DIFFERENT STATES:

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-----

```

STATE 1: CLASS DEALLOCATED, AVAILABLE

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
1 0 1 ADDRESS OF FIRST ENTRY 1
-----

```

STATE 2: POINTER TO FIRST ENTRY IN CLASS QUEUE

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
1 1 0 X1 SECURITY CODE 1 NUMBER OF PENDING REQS. 1
-----
(5 LSB OF ID SEG)
-----

```

STATE 3: CLASS ALLOCATED, NO ONE WAITING ON CLASS  
NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
1 1 1 X1 SECURITY CODE 1 NUMBER OF PENDING REQS. 1
-----

```

STATE 4: CLASS ALLOCATED, SOMEONE WAITING (SUSPENDED)  
NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

## CLASS QUEUE FORMAT:

WORD	CONTENTS
1	< LINKAGE WORD >
2	<T, CONTROL INFO, CODE >
3	<PRIORITY OF REQUESTOR > (CHANGED TO STATUS AT COMP.)
4	<TOTAL BLOCK LENGTH WORDS>
5	<CLASS ID WORD >
6	<USER BUFFER LENGTH > (CHANGED TO TLOG AT COMP.)
7	<OPTIONAL PARAMETER 1 >
8	<OPTIONAL PARAMETER 2 >
9	<WORD 1 OF USER BUFFER >
.	.
.	.
N+8	<WORD N OF USER BUFFER >

THE <T> FIELD (BITS 15-14 IN CONTROL WORD)  
IDENTIFIES THE REQUEST TYPE AS:

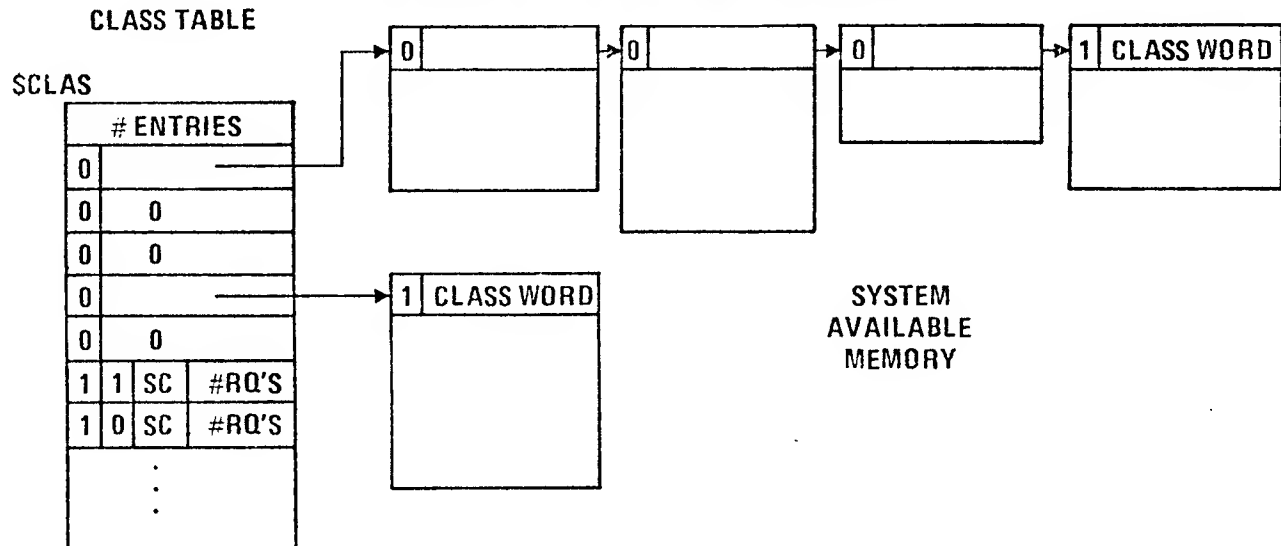
```

00 USER (NORMAL OPERATION)
01 USER (AUTOMATIC BUFFERING)
10 SYSTEM
11 CLASS I/O

```

SPECIFICATIONS: SCLAS = # ENTRIES IN CLASS TABLE  
HEADS CLASS TABLE

# CLASS I/O LINKING



sc = Security code (low 5 bits of owning program's index into keyword block)

#RQ'S = Number of outstanding class I/O segments (READ,WRITE, OR WRITE/READ).

## CLASS NUMBER

1. BITS 0-7: index into class table
2. BITS 8-12: programs keyword block index (see SC above)

## SYSTEM HANDLING OF CLASS I/O EVENTS

The action taken by RTE depends on the type of class I/O event and the state of the class queue listhead (or class table entry) as follows:

### CLASS I/O REQUESTS:

- STATE 1. STATE 3 IS SET UP, SECURITY CODE IS LOW 5 BITS OF PROGRAM ID NUMBER. COUNTER IS SET TO 1.
- STATE 2. THE COUNTER AT END OF QUEUE IS INCREMENTED BY 1
- STATE 3. THE COUNTER IS INCREMENTED BY 1.
- STATE 4. THE COUNTER IS INCREMENTED BY 1.

### ON COMPLETION OF CLASS I/O REQUESTS:

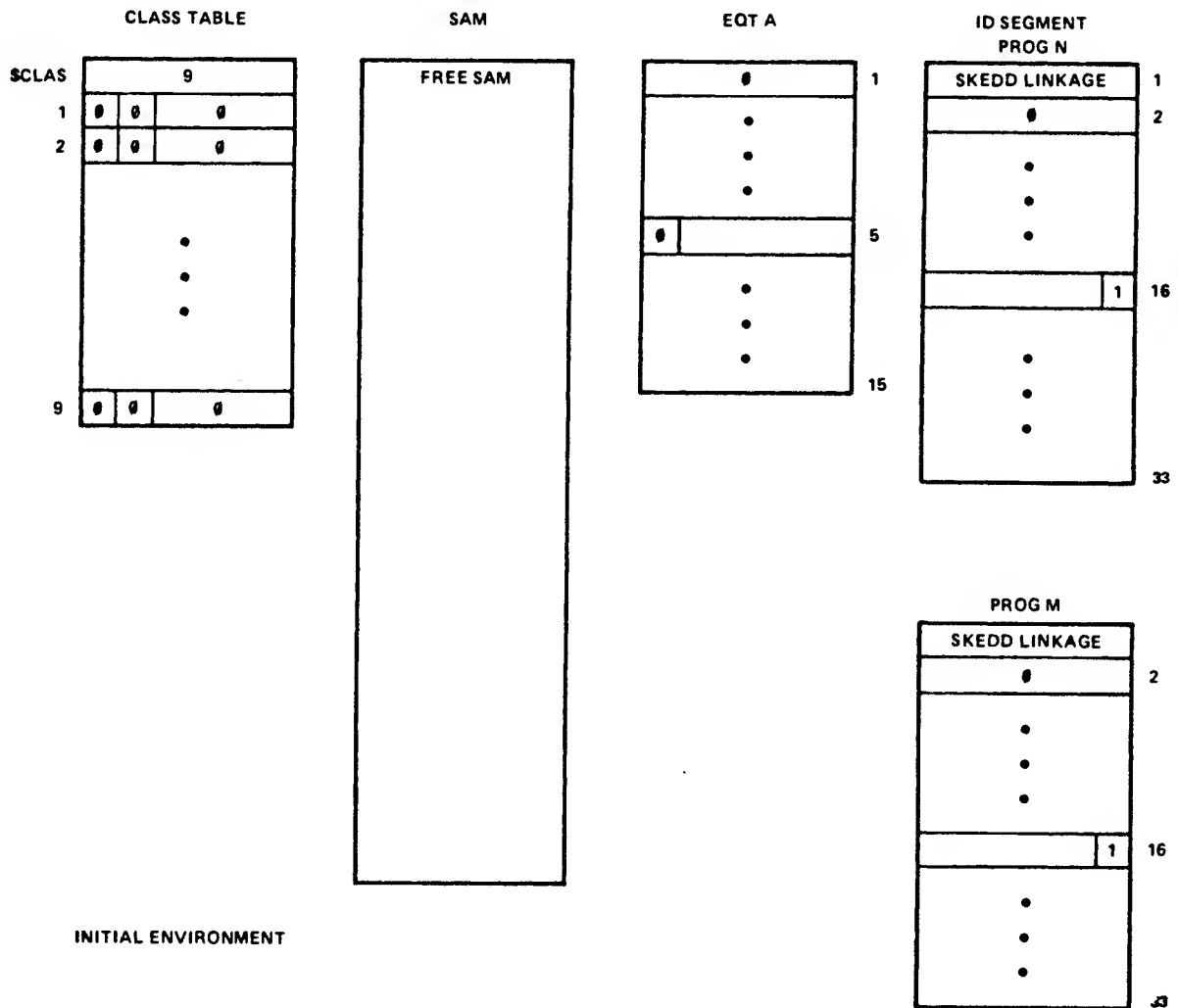
- STATE 1. ILLEGAL--SHOULD NEVER HAPPEN--BUFFER IS RETURNED AND THE COMPLETION IS IGNORED.
- STATE 2. THE NEW DATA IS ADDED AT THE END OF THE LIST (FIFO) AND THE COUNTER IS DECREMENTED BY 1.
- STATE 3. THE NEW DATA IS ADDED AT THE END OF THE LIST (FIFO) AND THE COUNTER IS DECREMENTED BY 1.
- STATE 4. THE WAITING PROGRAM IS SCHEDULED AND THE COUNTER IS DECREMENTED BY 1 AND THE SOMEONE WAITING BIT(BIT14) IS CLEARED.

### GET REQUESTS:

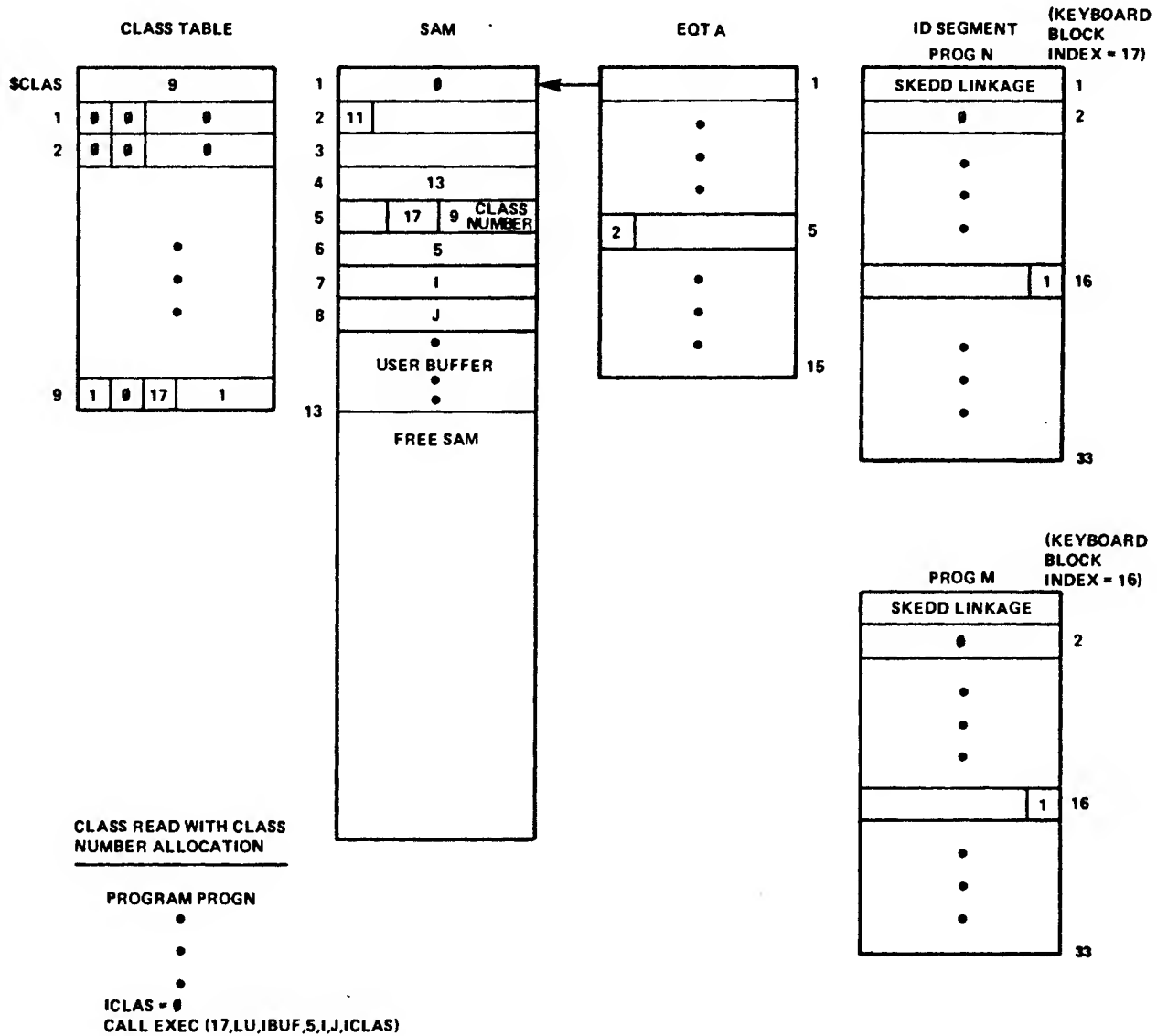
- STATE 1. ABORT THE PROGRAM IO00. NO CLASS.
- STATE 2. RETURN THE DATA FROM CLASS BUFFER
- STATE 3. SET THE SOMEONE WAITING BIT(BIT14). SUSPEND PROGRAM
- STATE 4. ABORT THE PROGRAM IO00. ONLY ONE PROGRAM MAY BE SUSPENDED PER CLASS.



# CLASS I/O EXAMPLE



# CLASS I/O EXAMPLE

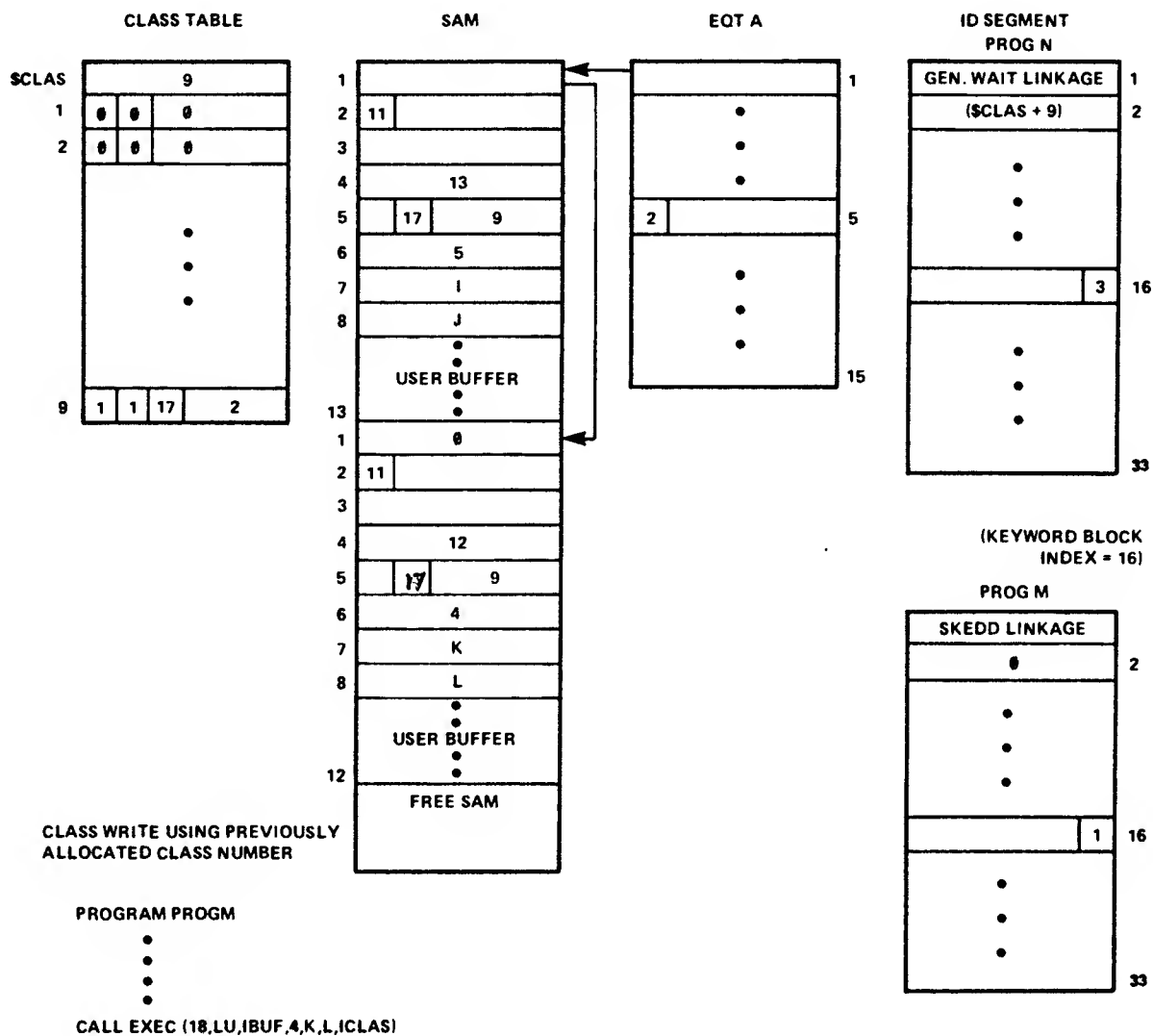


# CLASS I/O EXAMPLE

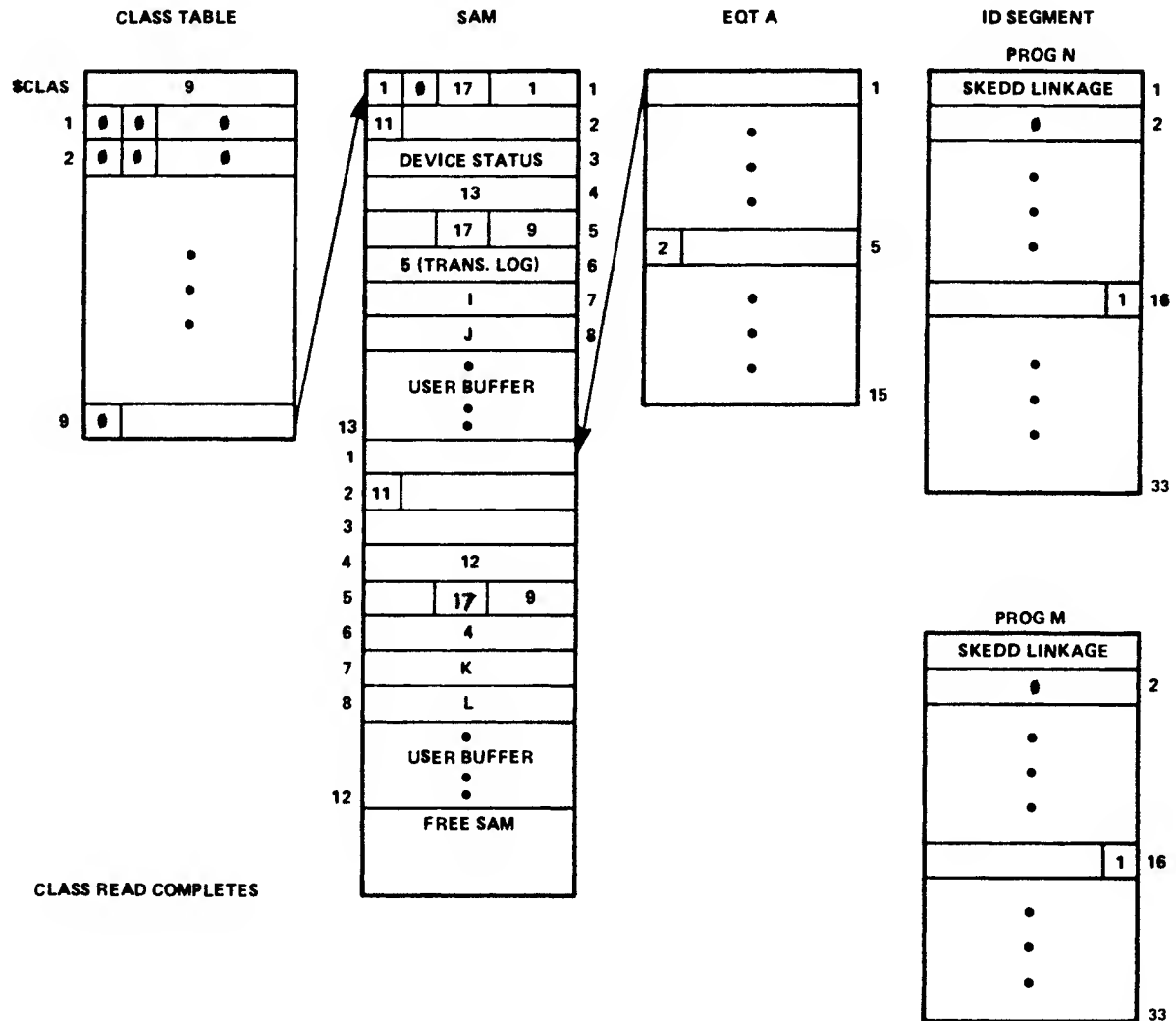


```
ICLAS = ICLAS + 60000B
CALL EXEC (21,ICLAS,IBUF,5,IRTN,JRTN,ICODE)
```

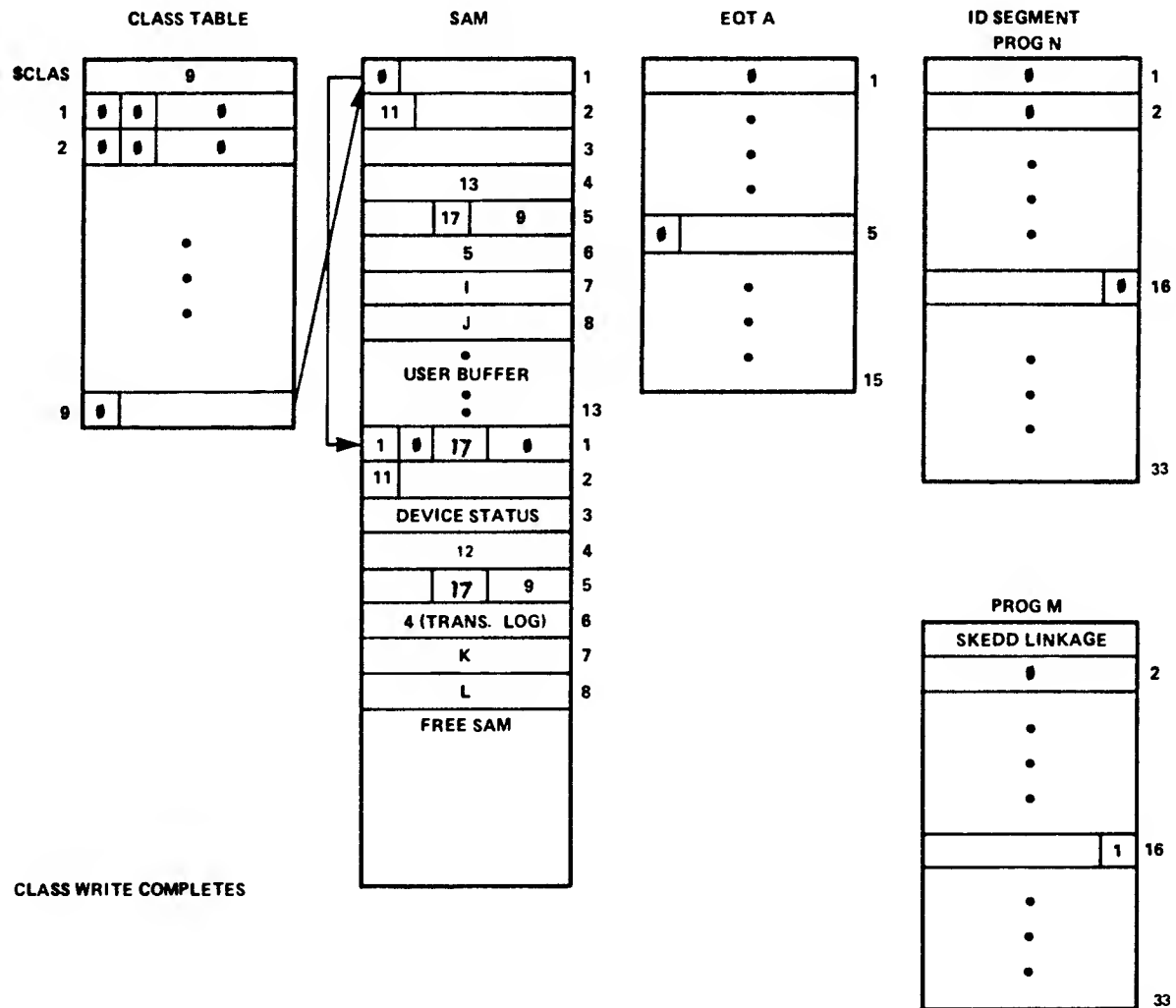
# CLASS I/O EXAMPLE



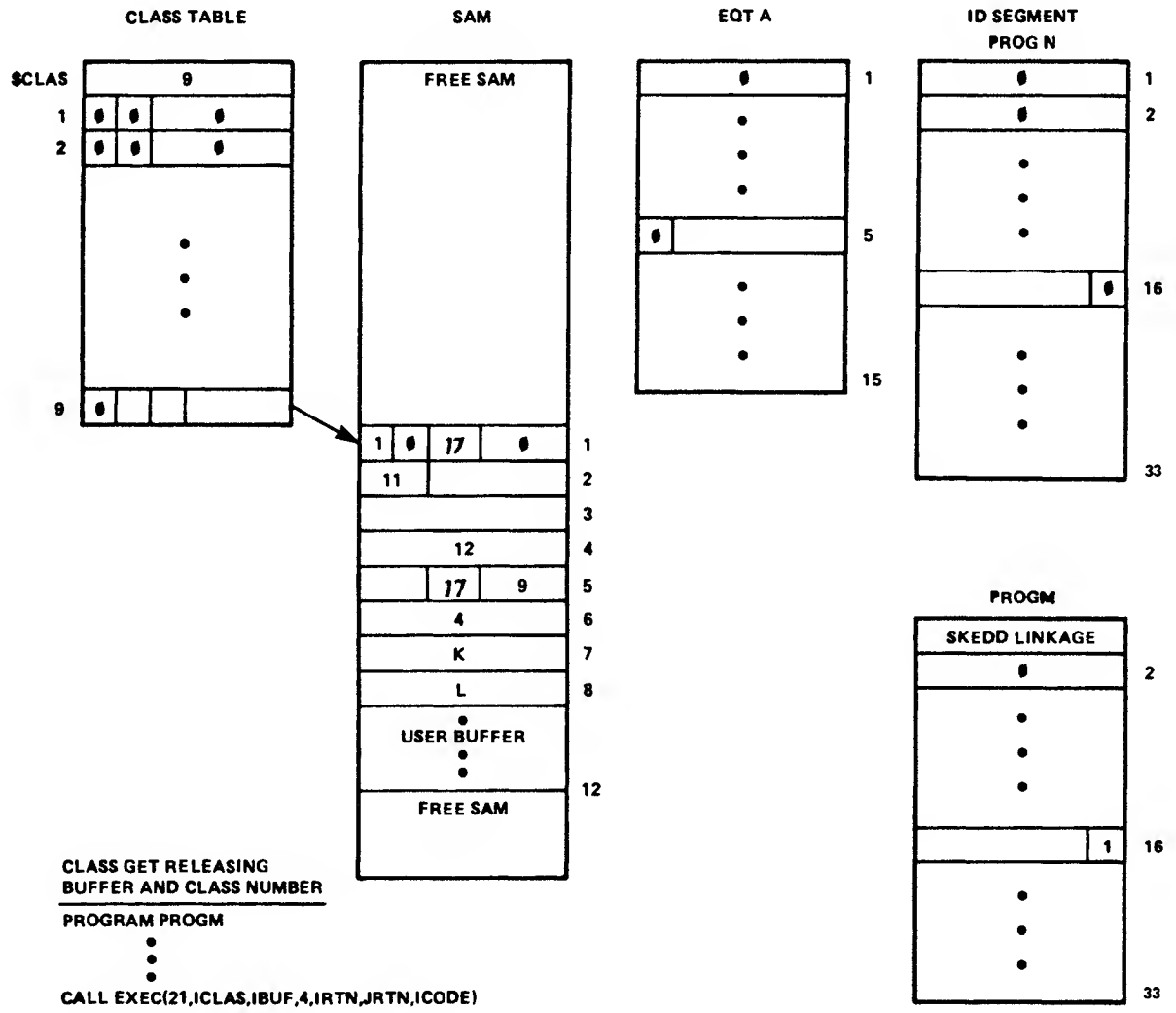
# CLASS I/O EXAMPLE



# CLASS I/O EXAMPLE



# CLASS I/O EXAMPLE



## CLASS I/O EXAMPLE



PROGRAM PROGM

•

•

•

**JCLAS = ICLAS + 20000B**

CALL EXEC (21,JCLAS,IBUF,4,KRTN,LRTN,ICODE)



RTE MODULES HANDLING  
CLASS I/O REQUESTS

Class I/O processing is handled in RTIOC in the following modules:

\$IORQ - Initiates class READ, WRITE, CONTROL, and READ/WRITE requests  
(EXEC 17, 18, 19, and 20)

IOCOM(\$CON1) - Unlinks the completed request from its EQT entry, checks for errors and branches to a subsection based on the I/O request type.

C.01(IOCOM) - Links the class request buffer into its class queue and reschedules the program, if any, waiting on the class number with a class GET request.

\$GTIO - Processes class GET requests (EXEC 21)

CLASS UTILITY PROGRAM\*

CLASS: CLASS TABLE IS AT 040227 WITH 12 ENTRIES!

CLASS: FOLLOWING COMMANDS ARE ACCEPTED:

DISPLAY,N1,N2,LU - DISPLAY STATUS OF CLASS TABLE FOR  
CLASS NUMBERS N1 THROUGH N2  
LIST,LU - LIST CONTENTS OF CLASS TABLE ON LU  
CLEAR - CLEAR OUT PENDING CLASS BUFFERS  
(CLASS NUMBER REQUESTED LATER)  
END - END

CLASS: TASK: 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.

CLASS POSSIBLE OWNERS SECU GET PROG OR BUFFER PRAMS  
#RQ SIZE OPT1 OPT2  
OCTAL

12	PRMPT	001	RSPNS
11	SMP	010	SPOUT
10	** AVAILABLE **		
9	** AVAILABLE **		
8	** AVAILABLE **		
7	** AVAILABLE **		
6	** AVAILABLE **		
5	** AVAILABLE **		
4	** AVAILABLE **		
3	** AVAILABLE **		
2	** AVAILABLE **		
1	** AVAILABLE **		

\* AVAILABLE IN LOCUS OR SOFTWARE SERVICE KIT

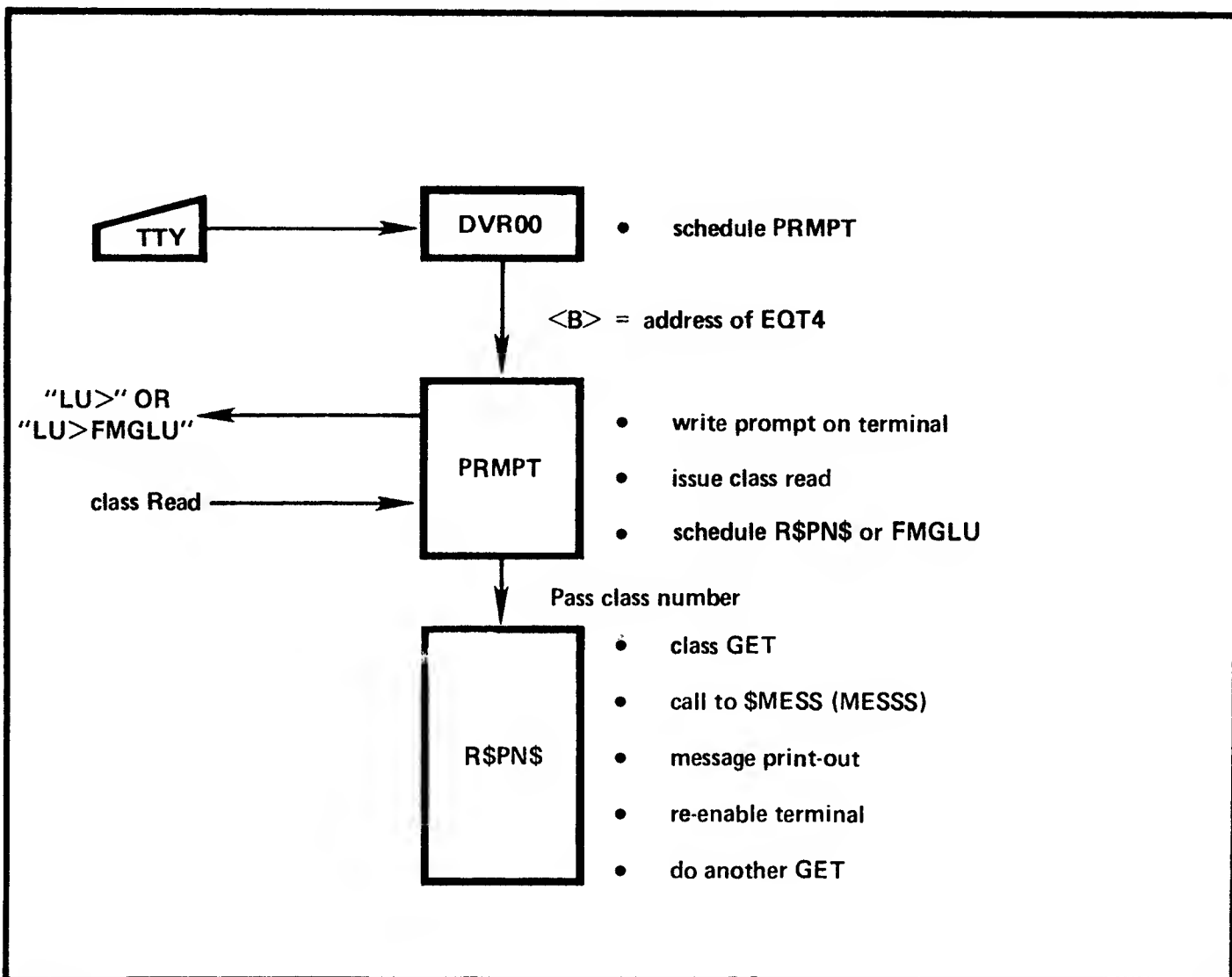
MULTI-TERMINAL MONITOR  
(MTM)



---

## ***MULTI-TERMINAL-MONITOR***

- **System Console Capability at Peripheral Keyboards**
- **PRMPT — schedule by interrupt**
- **R\$PN\$ — process command input**



## MTI FLOW

(Flow may also be easily followed in the listing.)

1. At generation: - Set "PRG,PRMPT" as interrupt table entry for each terminal.
2. At first call to each terminal ("CN,LU,20B", etc.) the terminal driver (DVR00,DVR05,etc.):
  - Moves PRMPT's ID segment number into the terminal's EQT entry.
  - Put the EQT entry address into the interrupt table.
  - Set EQT entry word 5 (bit 1) if the terminal was enabled.
3. Operator depresses a key and the terminal driver's continuation/completion section is entered thru \$CIC:
  - Checks that PRMPT is dormant.
  - Puts the address of EQT entry word 4 into ID segment word 11 (XB) of PRMPT.
  - Call \$LIST to schedule PRMPT

## MTM FLOW (cont'd)

### 4. PRMPT is entered:

- Calls library routine "TRMLU" (also checks for interactive LU-DVR00, 05, or 07) to get the terminal's LU from the address of EQT entry word 4.
- Check that LU and EQT are both up.
- If LU is locked, configure 9th parameter for EXEC I/O requests to write thru the LU lock. The parameter contains the RN# owner from the RN table and RN# from DRT.
- Disables the terminal with an EXEC 3 call. This inhibits PRMPT from being scheduled by the terminal's driver.
- Issue a zero-length record
- If FMGXX exists, schedule it, output "LU>FMGXX", and re-enable the terminal.
- If no FMGXX exists, output "LU> " and get the saved class number from \$MTM in Table Area I and issue a class READ on the terminal. Optional parameters are set to terminal LU and address of EQT entry word 4. Schedule R\$PN\$ immediately without wait (EXEC 10). It's probably in class GET suspend so scheduling errors are ignored.
- Terminate saving resources (pass the class number).

### 5. R\$PN\$ is entered:

- If R\$PN\$ was dormant, it picks up the class number passed in PRMPT's schedule request.
- +-- - Issues a class GET with wait saving the class number and releasing the class buffer.
- Like PRMPT, check for down LU, down EQT, or locked LU.
- Check for and process "FL", "BR", or "AB" commands.
- Calls \$MESS in SCHED to process the operator request.
- Prints out any system response from "\$MESS" with a class write.
- Enables the terminal with an EXEC 3 call. (re-checks LU in case it was re-assigned)
- +-- - Issue another class GET on the same class number.



## RE-ENTRANT PROCESSING



## RE-ENTRANT SUBROUTINES

### WHAT?

-----

Subroutines that do not modify their own instructions or local data and therefore may be called before completing its current task.

### WHY?

-----

Many executing programs can reference the same re-entrant subroutine on a priority basis (in RTE) and thus save memory space. A shared re-entrant subroutine can be used to manage a shared resource.

### HOW?

-----

Variable data associated with the re-entrant subroutine must be stored in separate private areas. This data normally includes arguments, return addresses, and temporary variables. Accomplished in RTE with \$LIBR,\$LIBX and temporary data blocks (TDB's).

### WHEN?

-----

In subroutines with execution times exceeding one millisecond. For shorter execution times, the overhead time RTE user in saving and restoring temporary data blocks (TDB's) makes the re-entrant structure unreasonable. In these cases, privileged subroutines should be used.

### WHERE?

-----

Normally in shared memory resident library subroutines which are only accessible by MR programs. (Type 6 or 14 programs in RTE). Re-entrant subroutines may also be shared by disc resident programs by placing them in SSGA (Type 30).

### WHAT NOT?

-----

Re-entrant subroutines in RTE may not call themselves (recursive).

# FORMAT OF RTE RE-ENTRANT SUBROUTINE

NAM ENTRY  
EXT \$LIBR,\$LIBX

ENTRY NOP Entry point of routine.  
JSB \$LIBR Call RTE-III to save temporary data.  
DEF TDB Address of temporary data.  
.  
.  
.  
.  
.  
.  
.  
.  
.  
EXIT JSB \$LIBX Call RTE-III to restore data.  
DEF TDB  
DEC m m is for routines with two return points in  
the calling program; 0 specifies the error-  
point return and 1 the normal return. For  
routines with only one return point, m = 0.  
TDB NOP System control word.  
0 = Subroutine not in use else;  
Points to word 2 of ID extension if TDB not in SAM.  
Points to original TDB location in re-entrant  
subroutine if TDB has been moved into SAM.  
DEC n+3 Total length of current block.  
NOP Return address to calling program.

T1  
.  
.....Temporary data (n words).  
.  
Tn

## LIMITATIONS:

1. Re-entrant subroutines cannot call themselves (recursion).
2. Re-entrant subroutines can call other re-entrant subroutines (and other privileged subroutines).

# MAKING A SUBROUTINE RE-ENTRANT UNDER RTE

Suppose we have a subroutine, SUB, with the calling sequence:  
`CALL SUB(A,B,C)`  
 and performs the operation:  
 $C = (A * 5) + (B * 3)$

Non re-entrant version of SUB:

	NAM	SUB,T	
	ENT	SUB	
	EXT	. ENTR	
A	NOP		
B	NOP		
C	NOP		
SUB	NOP		
	JSB	. ENTR	
	DEF	A	
	LDA	A,I	GET A
	MPY	D5	A*5
	STA	ATEMP	
	LDA	B,I	GET B
	MPY	D3	B*3
	ADA	ATEMP	(A+5) + (B+3)
	STA	C,I	SET C
	JMP	SUB,I	RETURN
	D3	DEC	3
	D5	DEC	5
	ATEMP	NOP	
		END	

To make SUB re-entrant:

	NAM	SUB,14	
	ENT	SUB	
	EXT	.ENTP,\$LIBR,\$LIBX	
TDB	NOP	(re-entrant data block)	
	DEC	1+3+3 (TDB length)	
RET	NOP	(return address)	
ATEMP	NOP	(local variables)	
A	NOP		
B	NOP	(argument addresses)	
C	NOP		
SUB	NOP		
	JSB	\$LIBR	(go re-entrant)
	DEF	TDB	
	JSB	.ENTP	(get argument addresses)
	DEF	A	(argument addresses in TDB)
	STA	RET	(save return address)
	LDA	A,I	
	MPY	D5	
	STA	ATEMP	
	LDA	B,I	
	MPY	D3	
	ADA	ATEMP	
	STA	C,I	
	JSB	\$LIBX	
	DEF	TDB	
	DEC	0	(return address adjustment)
	D3	DEC	3
	D5	DEC	5
		END	

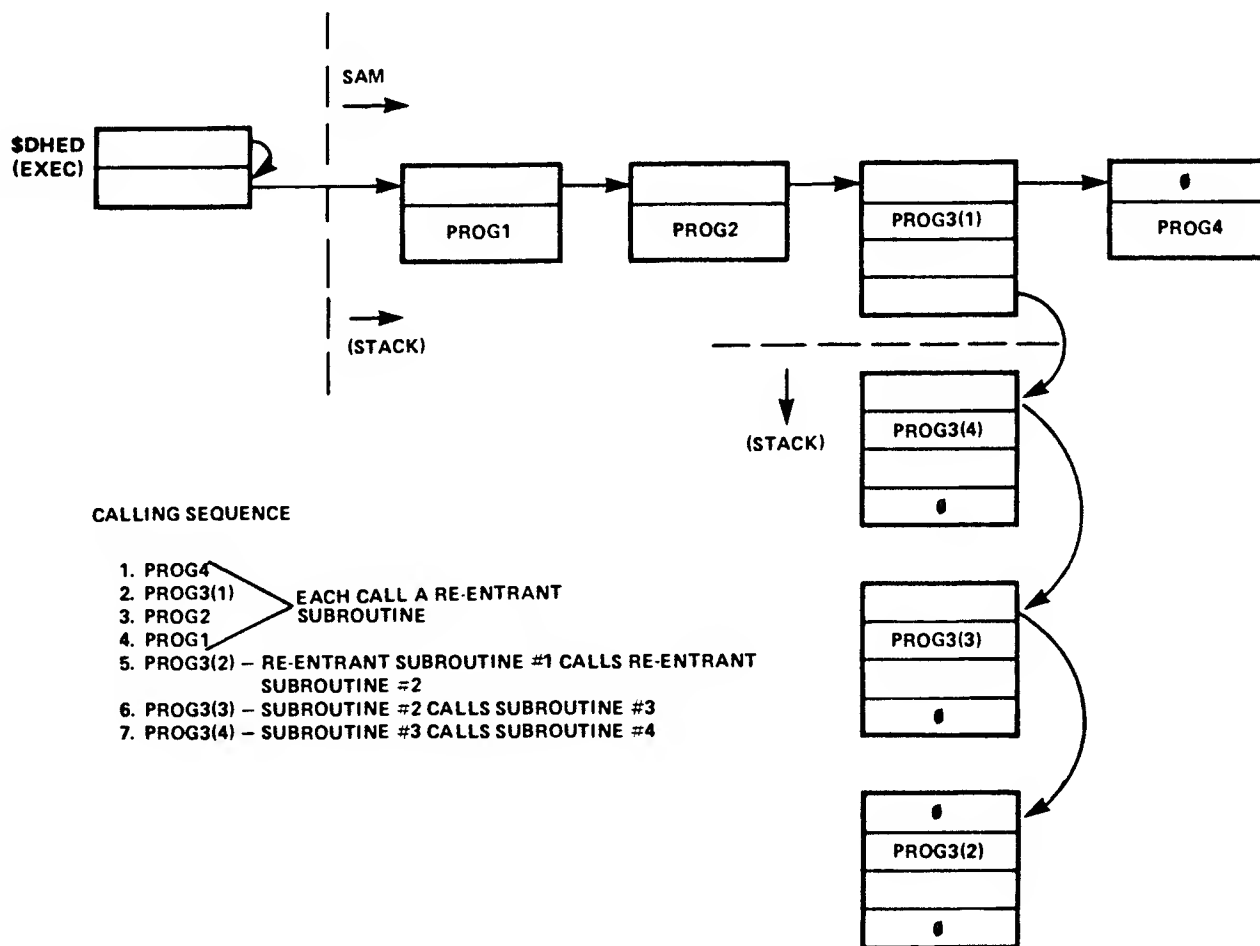
## \$LIBR/\$LIBX FUNCTIONS

1. \$LIBR - accepts requests to start a re-entrant or privileged subroutine.
2. \$LIBX - returns a user to normal processing after a re-entrant or privileged subroutine completes.
3. \$LIBR/\$LIBX message:
  - a. TDB's - when a re-entrant subroutine is re-entered \$LIBR moves the TDB into SAM. When the subroutine exits the SAM copy of the TDB is moved back into the subroutine's TDB.
  - b. Calling program's ID segment (word 21),  
 RE BIT: re-entrant subroutine is now in control  
 RM BIT: re-entrant memory (TDB in SAM) must be moved before program is dispatched.
  - c. Re-entrant table - an re-entrant table is created in SAM by \$LIBR each time a re-entrant call is made.

WORD	RE-ENTRANT TABLE FORMAT (listhead at \$DHED+1 in EXEC)
1	Link to next 4 word re-entrant table in SAM (0 = end of list)
2*	ID segment address of user program making re-entrant call
3	Pointer to TDB. Sign bit set if TDB has been moved to SAM.
4	Used to link re-entrant table of first re-entrant subroutine to a group of nested re-entrant subroutines.

\* Sign bit set if K+1 words of SAM allocated instead of K words as requested.

# RE-ENTRANT TABLE LIST FORMAT





## RE-ENTRANT PROCESSING EXAMPLE

1. PROGA calls re-entrant subroutine, SUB:

```
PROGRAM PROGA
.
.
.
.
.
CALL SUB(A,B,C)
.
.
.
END
```

2. PROGB also calls the re-entrant subroutine, SUB:

```
PROGRAM PROGB
.
.
.
.
CALL SUB(A,E,C)
.
.
.
END
```

3. SUB is a re-entrant subroutine which calls \$LIBR/\$LIBX

4. Sequence of execution will be:

- a. PROGA calls SUB
- b. While in SUB, PROGB begins execution and eventually calls SUB.
- c. PROGB exits SUB and terminates
- d. PROGA resumes execution in SUB and terminates.

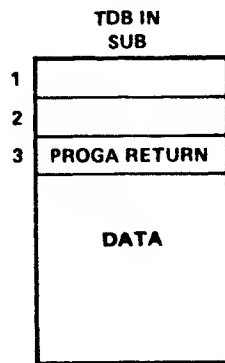
NOTE: Both PROGA & PROGB must be MR programs with SUB in the MR library. (Or SUB is in SSGA.)

RE-ENTRANT PROCESSING  
EXAMPLE FLOW

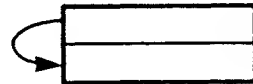
EXECUTION -----	NOTES -----
PROGA	PROGA starts execution.
CALL SUB(A,B,C)	SUB is called by PROGA.
MEMORY PROTECT	Since SUB is in MR library.
TRAP CELL 5(JSB CIC,I)	
\$CIC(RTIOC)	Detects MP violation on select code 5. (CIR=5)
\$RQST(EXEC)	Detect MP error and not DMS violation. Check that violating instruction was JSB or JSB,I. Determine that destination was MR library and calling program (PROGA) in MR.
LIBRC(EXEC-\$LIBR)	Detect that first word of TDB =0 and RE bit=0; therefore SUB is not being re-entered.
(\$ALC)	Allocate four word re-entrant table. Set up re-entrant table block and link into re-entrant list. Set RE bit in ID segment word 21. Set up words 1 and 3 of TDB.
\$RENT(DISPM)	Lower MP fence to start of MR library. Turn on memory protect and interrupt system. Restore SUB's registers and continue execution of SUB at "SUB NOP"+3. (Therefore "JSB \$LIBR" never executed!!!).
(The figure on 18-9 summarizes the state of tables/lists at this point)	
SUB	SUB continues execution.
PROGB	Begins execution before SUB completes.

# RE-ENTRANT PROCESSING TABLES/LISTS

MEMORY RESIDENT  
LIBRARY

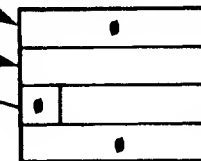


\$DHED



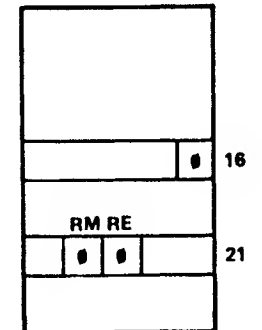
SAM

PROGA  
RE-ENTRANT TABLE

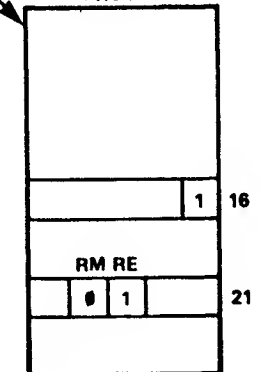


ID SEGMENTS

PROGB



PROGA



## EXECUTION

## NOTES

CALL SUB(A,B,C)

SUB is called by PROGB

MEMORY PROTECT

Since SUB is in MR library

SCIC(RTIOC)

Detect MP violaiton

SRQST(EXEC)

Determine that destination was MR library.

LIBRC(EXEC)

Make sure not a recursive call (TDB word 1,I = PROGBs ID segment address) Detect that TDB word 1 is not zero and therefore SUB is being re-entered. Allocate four word re-entrant table and buffer for copy of current TDB in SAM.

(\$ALC)

Set up re-entrant table.

(MTDB)

Copy current TDB into SAM and set RM bit in PROGA's ID segment.

Set RE bit in PROGB's ID segment.

Setup words 1 and 3 of TDB in SUB for PROGB's call.

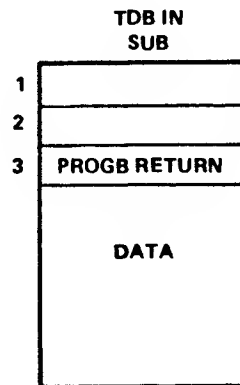
\$RENT(DISPM)

Continue execution of SUB (for PROGB's call) at "SUB NOP"+3.

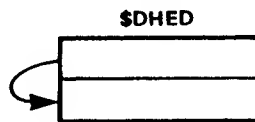
(See figure 18-11 for current system state)

# RE-ENTRANT PROCESSING TABLES/LISTS

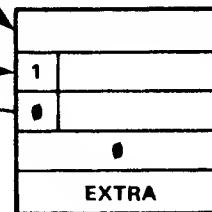
MEMORY RESIDENT  
LIBRARY



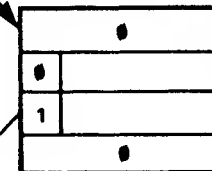
SAM



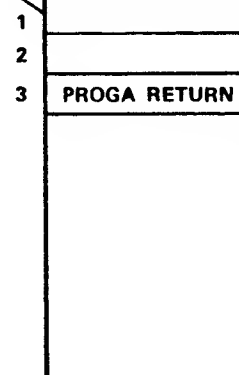
**PROGB  
RE-ENTRANT TABLE**



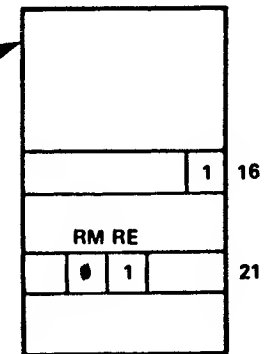
**PROGA  
RE-ENTRANT TABLE**



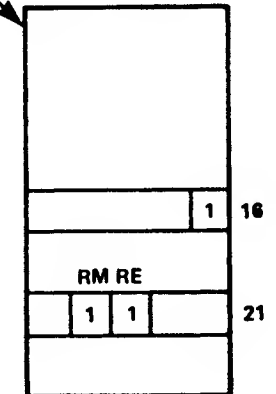
**SUB'S TDB  
FROM PROGA CALL**



ID SEGMENTS  
PROGB



**PROGA**



## EXECUTION

## NOTES

-----	-----
SUB	Complete execution from PROGB call
JSB \$LIBX	SUB calls RTE to terminate re-entrant processing
MEMORY PROTECT	
\$CIC	
\$RQST	Determines that destination was \$LIBX
LIBXC(EXEC) (\$RTN)	Clear RE bit in PROGB's ID segment. Unlink and return re-entrant table. Set word 1 of SUB's TDB to zero.
\$RENT	Continue execution of PROGB after SUB call.
PROGB	PROGB completes

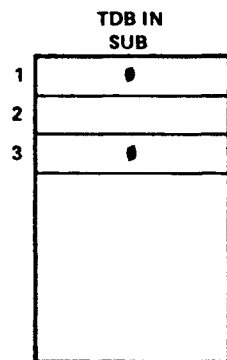
(See figure 18-13 for current system state)

SUB (PROGA)	PROGA is re-scheduled after its TDB is moved back from SAM.
-------------	---

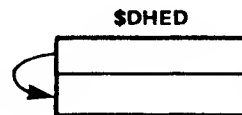
(See figure 18-9 for current system state)

# RE-ENTRANT PROCESSING TABLES/LISTS

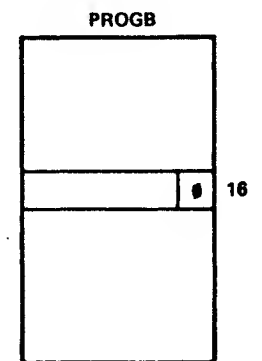
MEMORY RESIDENT  
LIBRARY



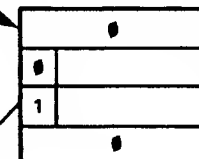
SAM



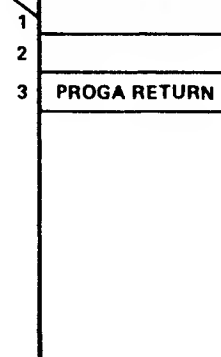
ID SEGMENTS



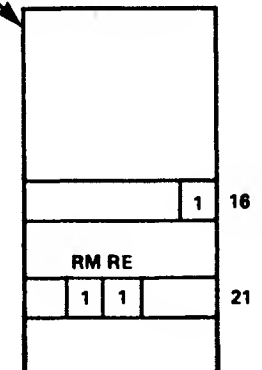
**PROGA  
RE-ENTRANT TABLE**



**SUB'S TDB  
FROM PROGA CALL**



**PROGA**



## RE-ENTRANT I/O

1. Any user program may do re-entrant I/O by issuing an I/O request from a re-entrant subroutine.
2. RTE processes a re-entrant I/O request as follows:
  - RTIOC notes that the RE bit in the ID segment is set and calls \$REIO in EXEC.
  - \$REIO\* verifies that the user's buffer is totally within a TDB and requests SAM for the TDB from \$ALC.
  - \$REIO moves the TDB into SAM and returns to RTIOC.
  - RTIOC puts the caller in I/O suspend (2) and calls LINK to link the request buffer on the EQT entry.
3. Note that the request buffer address in ID segment word 3 has the "L" bit set to indicate that the buffer is in SAM and not in the user's program.

\*Different from the utility subroutine, REIO.



## REIO UTILITY SUBROUTINE

CALL REIO(ICODE,ICNWD,IBUFR,IBUFL)

### REIO:

Sets up IBUFR in a TDB  
Goes re-entrant  
Issues the EXEC I/O request  
The I/O request is handled by RTE as re-entrant

### LIMITATIONS:

- Read/Write calls only
- Optional parameters not available
- Buffer (IBUFL) 129 words or less.
- Buffer address at least 5 words above the program's load point.

## PRIVILEGED SUBROUTINES

- Execute with interrupt system and memory protect turned off
- Used in programs that execute in less than one millisecond
- Format:

```
        NAM ENTRY,6
        EXT $LIBR,$LIBX

ENTRY  NOP          Entry points to the routine.
      JSB $LIBR      Call RTE-III to disable the interrupt
                    system and memory protect fence.
      NOP           Denotes privileged format.
      .
      .
      .
EXIT   JSB $LIBX      Call RTE-III to return to calling
                    program and enable interrupts and
                    memory protect fence.
      DEF ENTRY      Location of return address.
```

- Privileged routines may call other privileged routines but not re-entrant routines.

# .ZPRV/.ZRNT

The externals .ZPRV and .ZRNT are treated as "special" entry points in the RTE Disc-Based Operating Systems, in RTE-II, RTE-III and RTE-IV. The RTE Generator modifies the code that is loaded for subroutines that reference these externals, the changes made depend on whether or not the code is loaded into the core resident library (and hence may be shareable) or if the code is loaded with the program (not shareable), in the latter case the externals are satisfied by replacing the calls to .ZPRV or .ZRNT with an RSS (i.e., .ZPRV,RP,2001). These RP's are passed to the on-line loader in the same manner as an operators RP command at RTGEN time, thus, the on-line loader can perform the same functions as the RTE-Generator with respect to the externals .ZPRV, .ZRNT, \$LIBR, and \$LIBX. The following examples should help to illustrate how an assembled subroutine is modified.

NOTE - The capability of handling calls to REIO must also be added for compatibility reasons since the new library references this routine.

The code of .ENTP and .ENTR is included.

AS ASSEMBLED		WHEN "SUB" IN CORE RESIDENT LIBRARY		WHEN "SUB" NOT IN CORE RESIDENT LIBRARY	
-----					
-----					
N O R M A L   P R I V I L E G E D   R O U T I N E					
-----					
SUB	NOP	SUB	NOP	SUB	NOP
	JSB .ZPRV		JSB \$LIBR		RSS
	DEF LIBX		NOP		DEF LIBX
	...		...		...
	...		...		...
LIBX	JMP SUB,I	LIBX	JSB \$LIBX	LIBX	JMP SUB,I
	DEF SUB		DEF SUB		DEF SUB

.ZPRV/.ZRNT CALLING SEQUENCES

P R I V I L E G E D   W I T H   " . E N T R "

PARM1	NOP	PARM1	NOP	PARM1	NOP
PARM2	NOP	PARM2	NOP	PARM2	NOP
SUB	NOP	SUB	NOP	SUB	NOP
	JSB .ZPRV		JSB \$LIBF		RSS
	DEF LIBX		NOP		DEF LIBX
	JSB .ENTP		JSB .ENTP		JSB .ENTP
	DEF PARM1		DEF PRAM1		DEF PRAM1
	...		...		...
	...		...		...
LIBX	JMP SUB,I	LIBX	JSB \$LIBX	LIBX	JMP SUB,I
	DEF SUB		DEF SUB		DEF SUB

N O R M A L   R E - E N T R A N T   R O U T I N E

SUE	NOP	SUB	NOP	SUB	NOP
	JSB .ZRNT		JSB \$LIBR		RSS
	DEF LIBX		DEF TDB		DEF LIBX
	...		...		...
	...		...		...
	ISZ SUB		ISZ SUB		ISZ SUB
	ISZ TDB+2		ISZ TDB+2		ISZ TDB+2
	NOP		NOP		NOP
	...		...		...
LIBX	JMP SUB,I	LIBX	JSB \$LIBX	LIBX	JMP SUB,I
	DEF TDB		DEF TDB		DEF TDB
	DEC 0		DEC 0		DEC 0

R E - E N T R A N T   W I T H   " . E N T R "

PRAM1	NOP	PRAM1	NOP	PRAM1	NOP
PRAM2	NOP	PRAM2	NOP	PRAM2	NOP
SUB	NOP	SUB	NOP	SUB	NOP
	JSB .ZRNT		JSB \$LIBF		RSS
	DEF LIBX		DEF TDB		DEF LIBX
	JSB .ENTP		JSB .ENTP		JSB .ENTP
	DEF PRAM1		DEF PRAM1		DEF PRAM1
	STA TDB+2		STA TDB+2		STA TDB+2
	...		...		...
	...		...		...
LIBX	JMP TDB+2,I	LIBX	JSB \$LIBX	LIBX	JMP TDB+2,I
	DEF TDB		DEF TDB		DEF TDB
	DEC 0		DEC 0		DEC 0

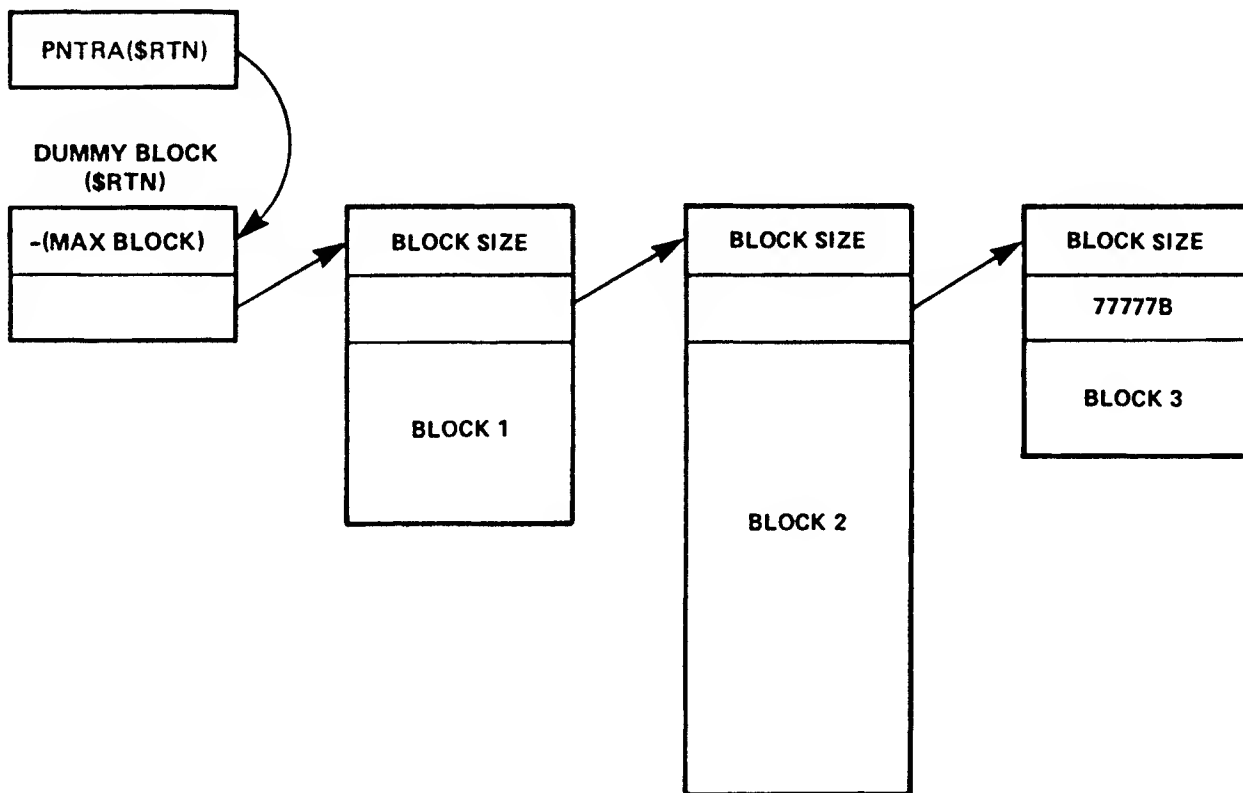
SYSTEM AVAILABLE MEMORY  
(SAM)



## SAM MANAGEMENT

- RTE-IV generator/configurator sets in base page EQT1 thru EQT6 the address and size of each of the 3 possible pieces of SAM. (SAM default block, SAM extension, and SAM in Table Area I.)
- At boot-up \$STRT maps SAM in contiguous map register allowing up to 5 bad pages in SAM.
- \$STRT links each piece of SAM into the SAM free list.
- PNTRA in \$ALCM is the free SAM listhead.
- \$ALC in \$ALCM allocates requested contiguous blocks of free SAM.
- \$RTN (\$ALCM) returns blocks to the free SAM list.
- As blocks of SAM are returned, only the highest priority program on the memory suspend list (4) is checked for scheduling (1).

# FREE SAM LIST





## 4ALC/\$RTN

### \$ALC CALLING SEQUENCE:

(P) JSB \$ALC  
(P+1) (# OF WORDS NEEDED)  
(P+2) -RETURN NO MEMORY EVER (A)=-1, (B)=MAX EVER  
(P+3) -RETURN NO MEMORY NOW (A)=0, (B)=MAX NOW  
(P+4) -RETURN OK (A)=ADDR, (B)=SIZE OR SIZE+1

TO FIND OUT HOW LARGE A BLOCK MAY EVER BE ALLOCATED:

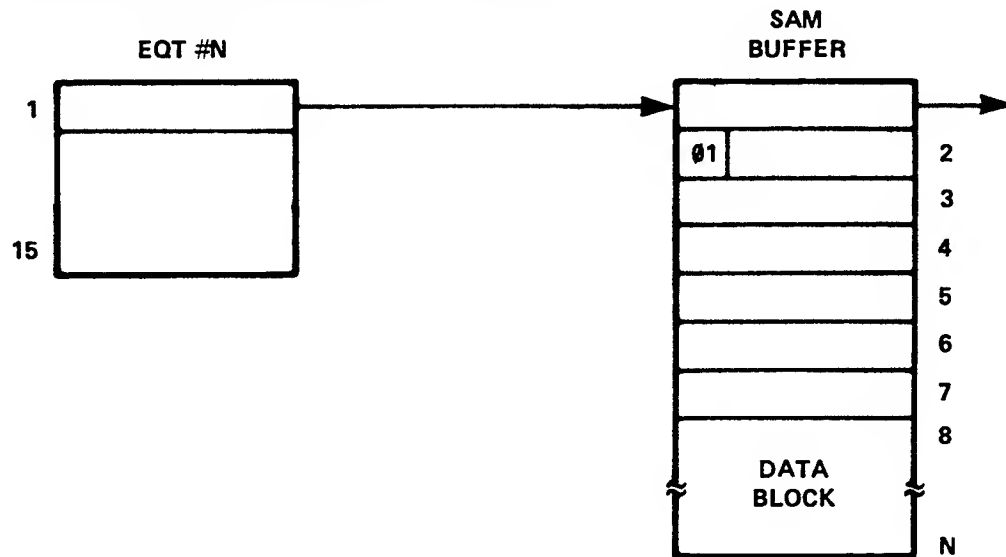
JSB \$ALC  
DEC 32767

### \$RTN CALLING SEQUENCE:

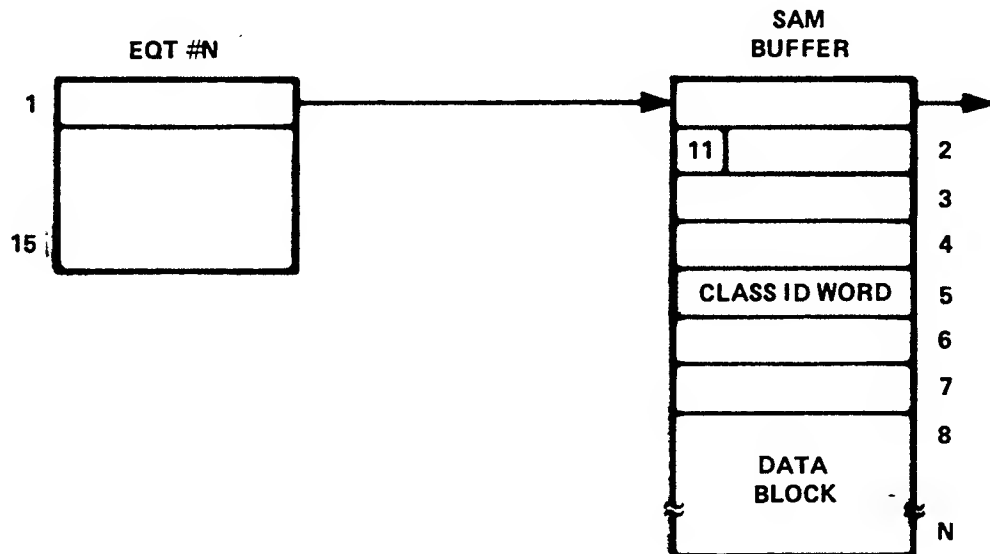
(P) JSB \$FTN  
(P+1) (FWA OF BUFFER)  
(P+2) (# OF WORDS RETURNED)  
(P+3) -RETURN- (ALL REGISTERS DESTROYED)

# SAM USERS

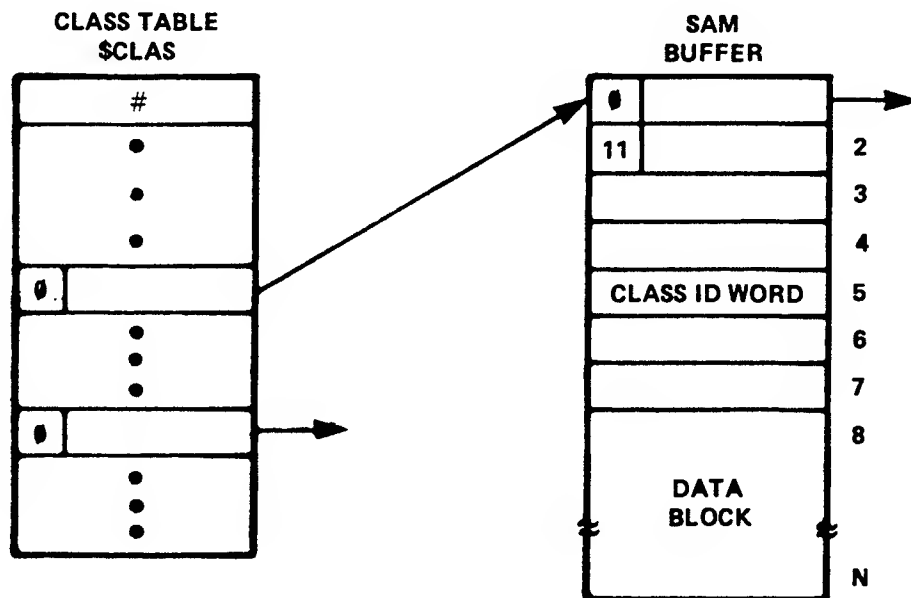
## I. AUTOMATIC OUTPUT BUFFERING I/O REQUESTS



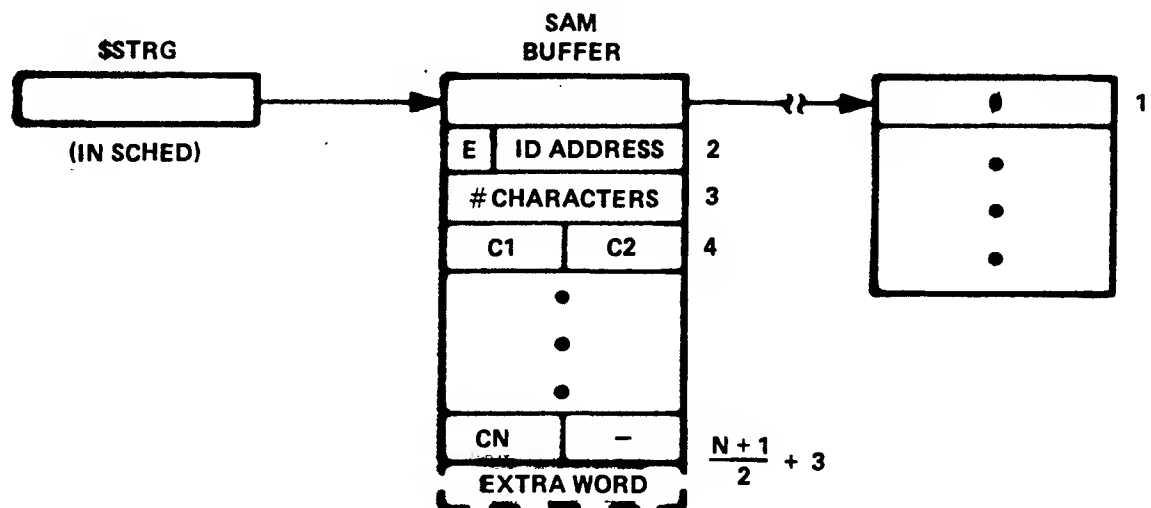
## II. CLASS I/O REQUESTS



### III. CLASS QUEUE



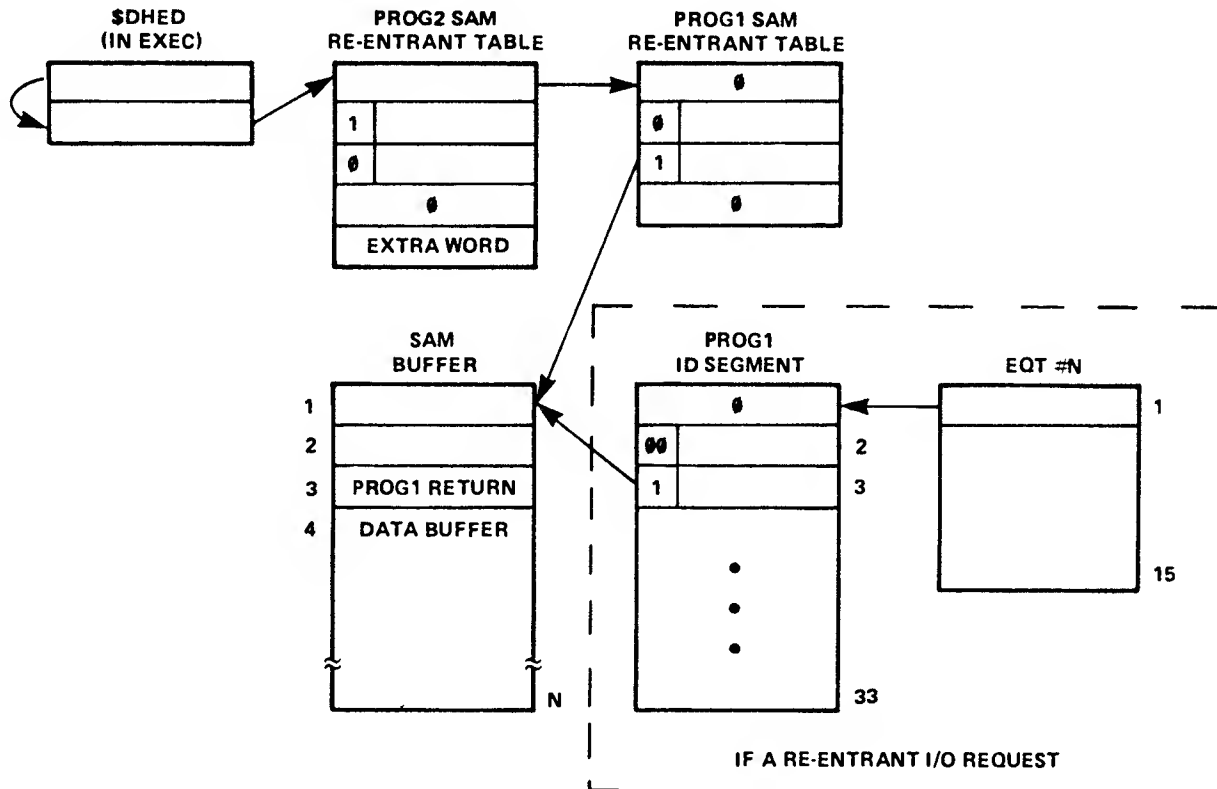
### IV. PARAMETER STRINGS FROM "ON" OR "RU" COMMANDS



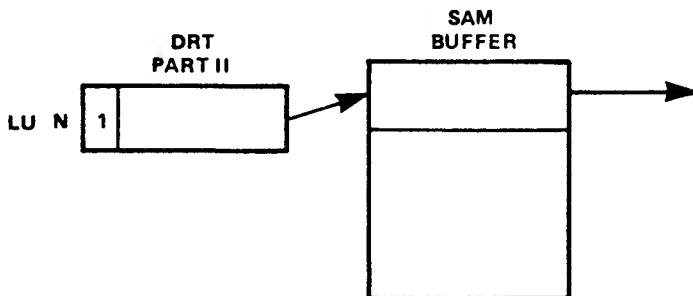
E = 1 IF \$ALC HAD TO ALLOCATE AN EXTRA WORD IN SAM

## V. RE-ENTRANT PROCESSING

- a. ID EXTENSIONS CREATED EACH TIME A RE-ENTRANT ROUTINE IS CALLED.
- b. MOVED TDB'S (TEMPORARY DATA BLOCK) BECAUSE OF RE-ENTRANT SUBROUTINES BEING RE-ENTERED OR AN I/O REQUEST FROM A RE-ENTRANT ROUTINE WITH THE REQUEST BUFFER IN THE TDB.



## VI. DOWN DEVICE BUFFERS



## SAM USAGE PERMITS

- I/O WITHOUT WAIT
- PROGRAM TO PROGRAM COMMUNICATION
- PROGRAM SWAPPING WHILE PROGRAM WAITS FOR I/O COMPLETION
- A MEMORY POOL USEABLE BY MULTIPLE USERS ON AN "AS NEEDED" BASIS



## I/O DRIVERS





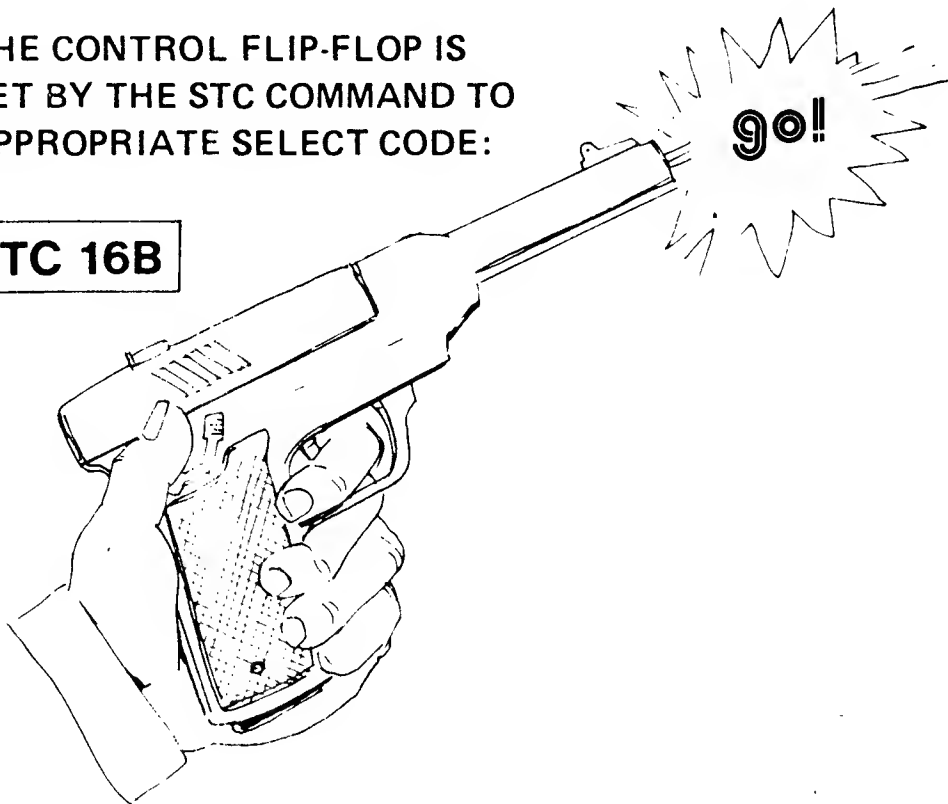
## WHAT IS A DRIVER?

A DRIVER IS A ROUTINE WHICH RESIDES IN THE SYSTEM AND IS RESPONSIBLE FOR ALL DATA TRANSFERRED BETWEEN THE I/O DEVICE AND COMPUTER DURING THE INITIATION, CONTINUATION, AND COMPLETION PHASES OF I/O.

# HOW IS A DEVICE STARTED?

- \* BY A CONTROL FLIP-FLOP ON THE INTERFACE CARD.
- \* WHEN SET, THE CONTROL FLIP-FLOP GENERATES A START COMMAND.
- \* THE DEVICE THEN PERFORMS ONE OPERATION CYCLE.
- \* THE CONTROL FLIP-FLOP IS SET BY THE STC COMMAND TO APPROPRIATE SELECT CODE:

**STC 16B**



# WHAT HAPPENS TO DATA ?

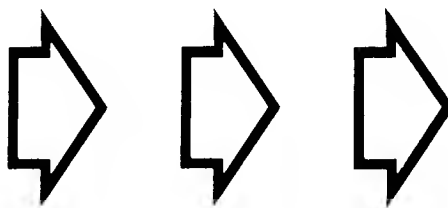
- ★ THE INTERFACE CARD HAS A STORAGE BUFFER.

- ★ TO INPUT DATA TO THE PROCESSOR FROM THE BUFFER USE:

**LIA 16B**

- ★ TO OUTPUT DATA FROM THE PROCESSOR TO THE BUFFER USE:

**OTA 16B**



# WHEN IS DATA READY ?

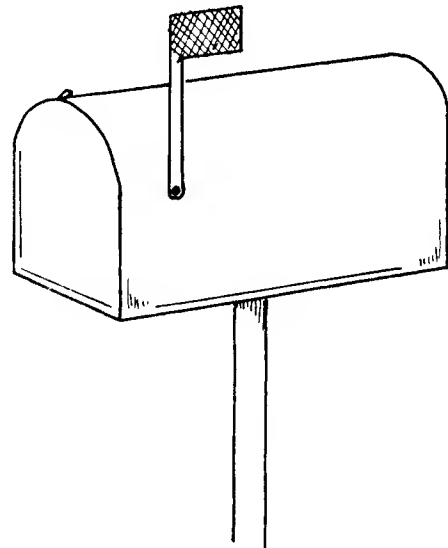
- WHEN THE DEVICE CYCLE IS COMPLETE, A FLAG FLIP-FLOP IS SET ON THE I/O CARD.

TO CHECK THE FLAG FLIP-FLOP USE

**SFS 16B**

- IF THE FLAG IS ALREADY SET – CLEAR THE FLAG BEFORE STARTING AN OPERATION.
- TO ELIMINATE TIMING CONFLICTS USE “SET CONTROL” AND “CLEAR FLAG” IN THE SAME COMMAND.

**STC 16B, C**



# AN INPUT AND OUTPUT EXAMPLE

ASSUME SELECT CODE 16 FOR INPUT AND 17  
FOR OUTPUT:

## INPUT

STC	16B,C	CLEAR FLAG AND START OPERATION
SFS	16B	OPERATION FINISHED ?
JMP	*-1	NO, KEEP CHECKING
LIA	16B	YES, GET DATA

## OUTPUT

OTA	17B	OUTPUT DATA TO I/O CARD
STC	17B,C	CLEAR FLAG AND START OPERATION
SFS	17B	FINISHED ?
JMP	*-1	NO, KEEP CHECKING
	•	YES, CONTINUE
	•	
	•	

# ASYNCHRONOUS DRIVERS

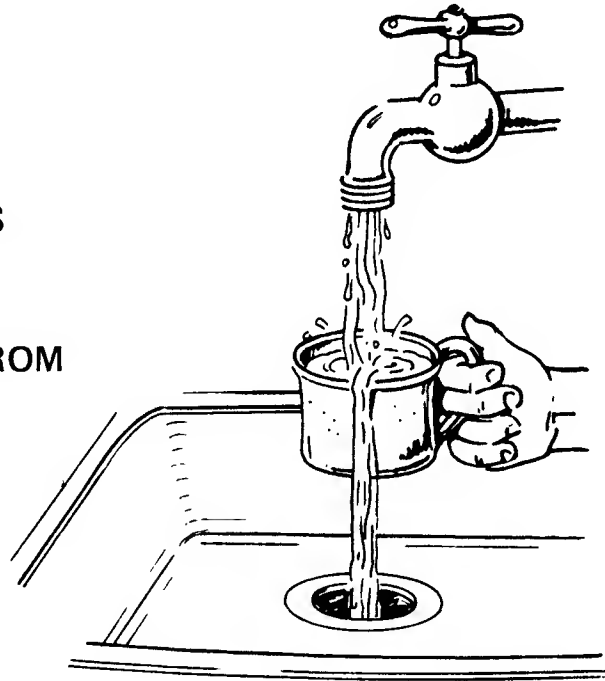
- ASYNCHRONOUS DEVICES MAKE ONE DATA TRANSFER AND WAIT FOR COMPUTER ACKNOWLEDGEMENT (HANDSHAKE).



- DATA IS TRANSFERRED ONLY AS FAST AS THE DRIVER REQUESTS IT. (CRT TERMINAL OR LINE PRINTER)

# SYNCHRONOUS DRIVERS

- ▶ SYNCHRONOUS DEVICES  
DO NOT WAIT FOR  
DATA TRANSFER  
ACKNOWLEDGEMENT FROM  
THE COMPUTER.



- ▶ IF THE DRIVER DOESN'T  
KEEP UP, DATA IS LOST.

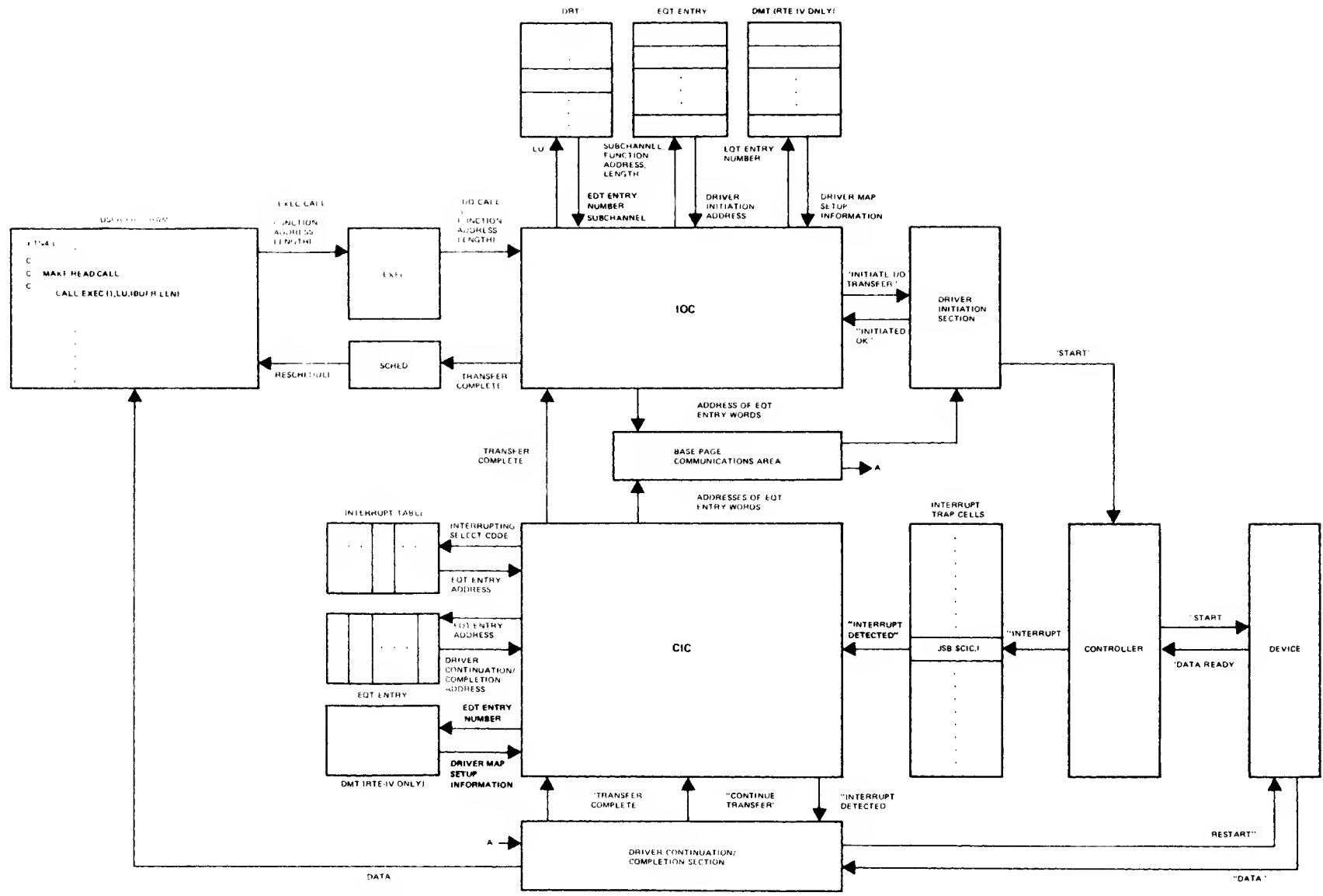
- ▶ SINCE THE DEVICE KEEPS RUNNING ONCE IT HAS BEEN  
STARTED, DCPC SHOULD BE USED.  
(MAGNETIC TAPE OR DISC)

## STANDARD RTE DRIVERS

- TWO SECTIONS
  - INITIATION
  - CONTINUATION/COMPLETION
  
- INTERFACE WITH SYSTEM MODULES
  - \$CIC
  - IOC
  
- USE DATA AREAS
  - EQUIPMENT TABLE
  - SYSTEM BASE PAGE COMMUNICATION AREA
  
- NORMALLY OPERATE WITH INTERRUPT SYSTEM OFF



# UNBUFFERED EXEC READ REQUEST FLOW

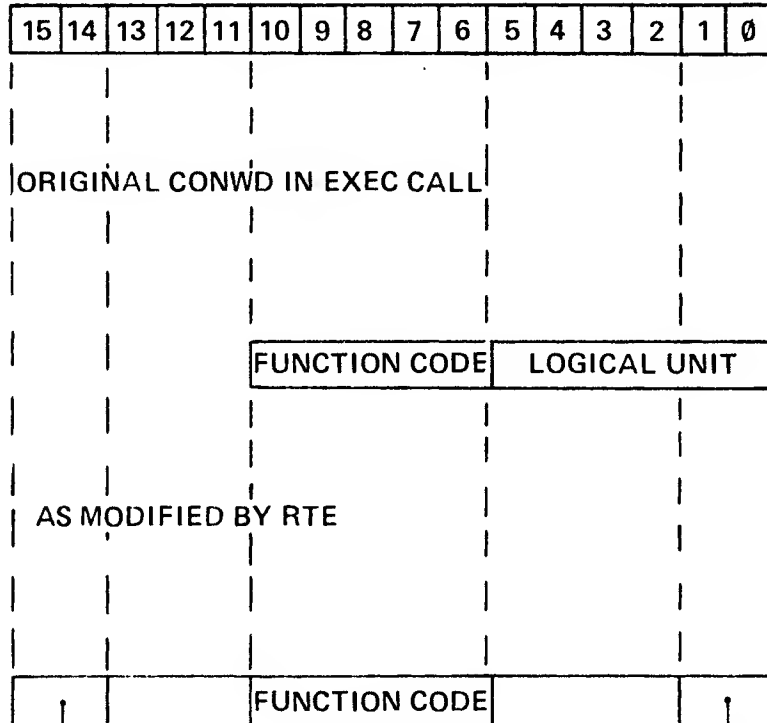


# EQT

WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O REQUEST LIST POINTER <C>														
2	R	DRIVER "INITIATION" SECTION ADDRESS <A>														
3	R	DRIVER "CONTINUATION/COMPLETION" SECTION ADDRESS <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	SUBCHANNEL #<C>					I/O SELECT CODE # <A>					
5	AV <F>		EQUIPMENT TYPE CODE <A>					STATUS <E>								
6	CONWD (CURRENT I/O REQUEST WORD) <C>															
7	REQUEST BUFFER ADDRESS <C>															
8	REQUEST BUFFER LENGTH <C>															
9	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
10	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
11	TEMPORARY STORAGE FOR DRIVER <D>															
12	TEMPORARY STORAGE FOR DRIVER <D> OR EQT EXTENSION SIZE, IF ANY <A>															
13	TEMPORARY STORAGE FOR DRIVER <D> OR EQT EXTENSION STARTING ADDRESS, IF ANY <A>															
14	DEVICE TIME-OUT RESET VALUE <B>															
15	DEVICE TIME-OUT CLOCK <C>															
WHERE THE LETTERS IN BRACKETS (<>) INDICATE THE NATURE OF EACH DATA ITEM, AS FOLLOWS:																
<A> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); NEVER CHANGES.																
<B> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); CAN BE CHANGED ON-LINE.																
<C> = SET UP OR MODIFIED AT EACH I/O INITIALIZATION.																
<D> = AVAILABLE FOR USE AS TEMPORARY STORAGE BY DRIVER.																
<E> = CAN BE SET BY DRIVER.																
<F> = MAINTAINED BY SYSTEM.																
AND WHERE:																
R = (RESERVED FOR SYSTEM USE)																
I/O REQUEST LIST POINTER = POINTER TO LIST OF REQUESTS QUEUED UP ON THIS EQT ENTRY. FIRST ENTRY IN LIST IS CURRENT REQUEST IN PROGRESS; ZERO IF NO REQUESTS.																
D = 1 IF DCPC REQUIRED																
B = 1 IF AUTOMATIC OUTPUT BUFFERING USED																
P = 1 IF DRIVER IS TO PROCESS POWER FAIL																
S = 1 IF DRIVER IS TO PROCESS TIME-OUT																
T = 1 IF DEVICE TIMED OUT (SYSTEM SETS TO ZERO BEFORE EACH I/O REQUEST)																
SUBCHANNEL # = LAST SUBCHANNEL ADDRESSED																

# STRUCTURE OF EQT WORD #6 AS SET BY RTE

BIT POSITION



→ 00 = STANDARD CALL  
 01 = BUFFERED CALL  
 11 = CLASS CALL

→ 01 = READ CALL  
 10 = WRITE CALL  
 11 = CONTROL CALL

# BASE PAGE COMM. AREA CONTENTS

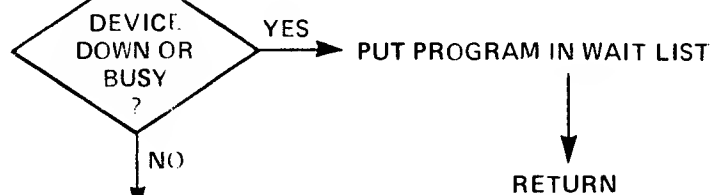
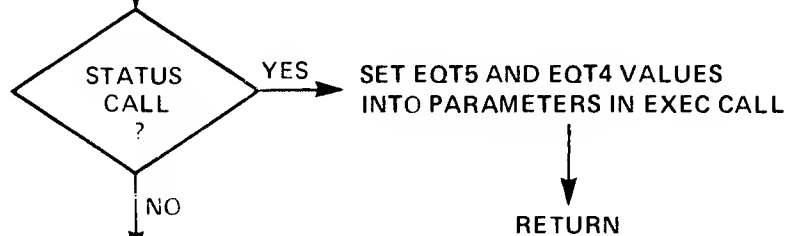
OCTAL LOCATION	CONTENTS	DESCRIPTION
• • •		
01650	EQTA	Address of Equipment Table (EQT)
01651	EQT#	Number of EQT entries
01652	DRT	Address of Device Reference Word 1 Table
01653	LUMAX	Number of logical units (in Device Reference Table)
01654	INTBA	Address of Interrupt Table
01655	INTLG	Number of Interrupt Table entries
01656	TAT	Address of Track Assignment Table (disc-based systems only)
01657	KEYWD	Address of keyword block
01660 01661 01662 01663 01664 01665 01666 01667 01670 01671 01672	EQT1 EQT2 EQT3 EQT4 EQT5 EQT6 EQT7 EQT8 EQT9 EQT10 EQT11	} Addresses of first 11 words of current EQT entry (see location of 01771 for last 4 words)
01673	CHAN	Current DCPC Select Code (6 or 7)
• • •		
01717	XEQT	ID segment address of current program
• • •		
01737	DUMMY	I/O channel of privileged interrupt card (0 if none)
• • •		
01770	MPTFL	Memory Protect On/Off (0/1) flag.
01771 01772 01773 01774	EQT12 EQT13 EQT14 EQT15	} Addresses of last 4 words of current EQT entry

# WHAT HAPPENS WHEN A PROGRAM CALLS YOUR DRIVER?

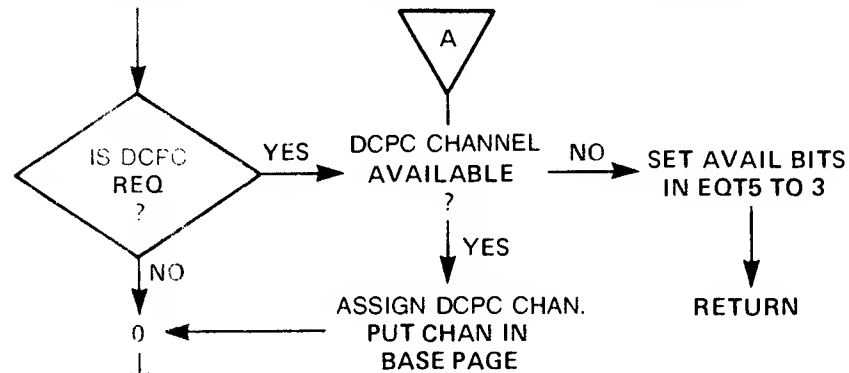
AN EXEC CALL IS MADE TO YOUR LU (CAUSES MEM. PROTECT VIOLATION)

THE RTE EXEC IS ENTERED AND INPUT-OUTPUT CONTROL (IOC) CALLED

THE LU IS TRACED TO AN EQT AND ALL 15 ADDRESSES PUT IN BASE PAGE



SYSTEM ROUTINE DRVR CALLED, A & B REGISTERS MEANINGLESS

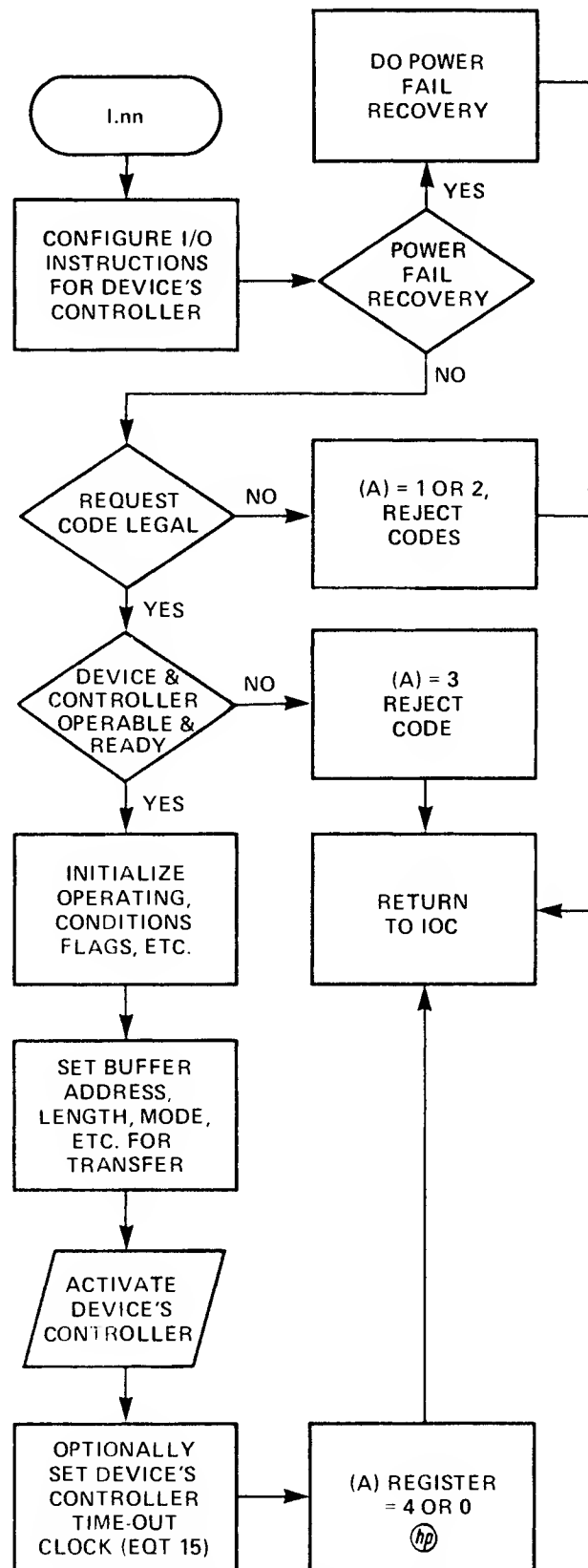


MAP DRIVER INTO DRIVER PARTITION (UNLESS IN SDA) USING THE DRIVER MAPPING TABLE. PASS ALL PARAMETERS TO EQT, SET EQT14 INTO EQT15 TO START TIMEOUT CLOCK AND CLEAR TIME-OUT BIT IN EQT4

LOAD A REG WITH SELECT CODE FROM EQT4, LOAD B WITH INITIATOR ADDRESS

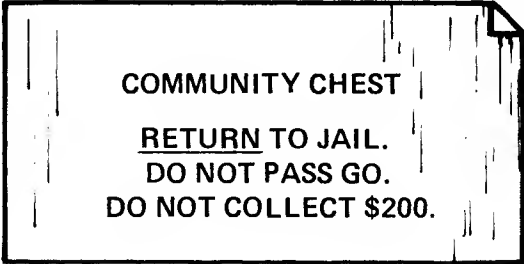
CALL INITIATOR BY DOING JSB B, I (SYSTEM MAP)  
OR UJP, I (USER MAP)

# RTE DRIVER INITIATION SECTION



hp IF A = 4 SET B = TRANSMISSION LOG

# DRIVER RULES FOR INITIATOR RETURN

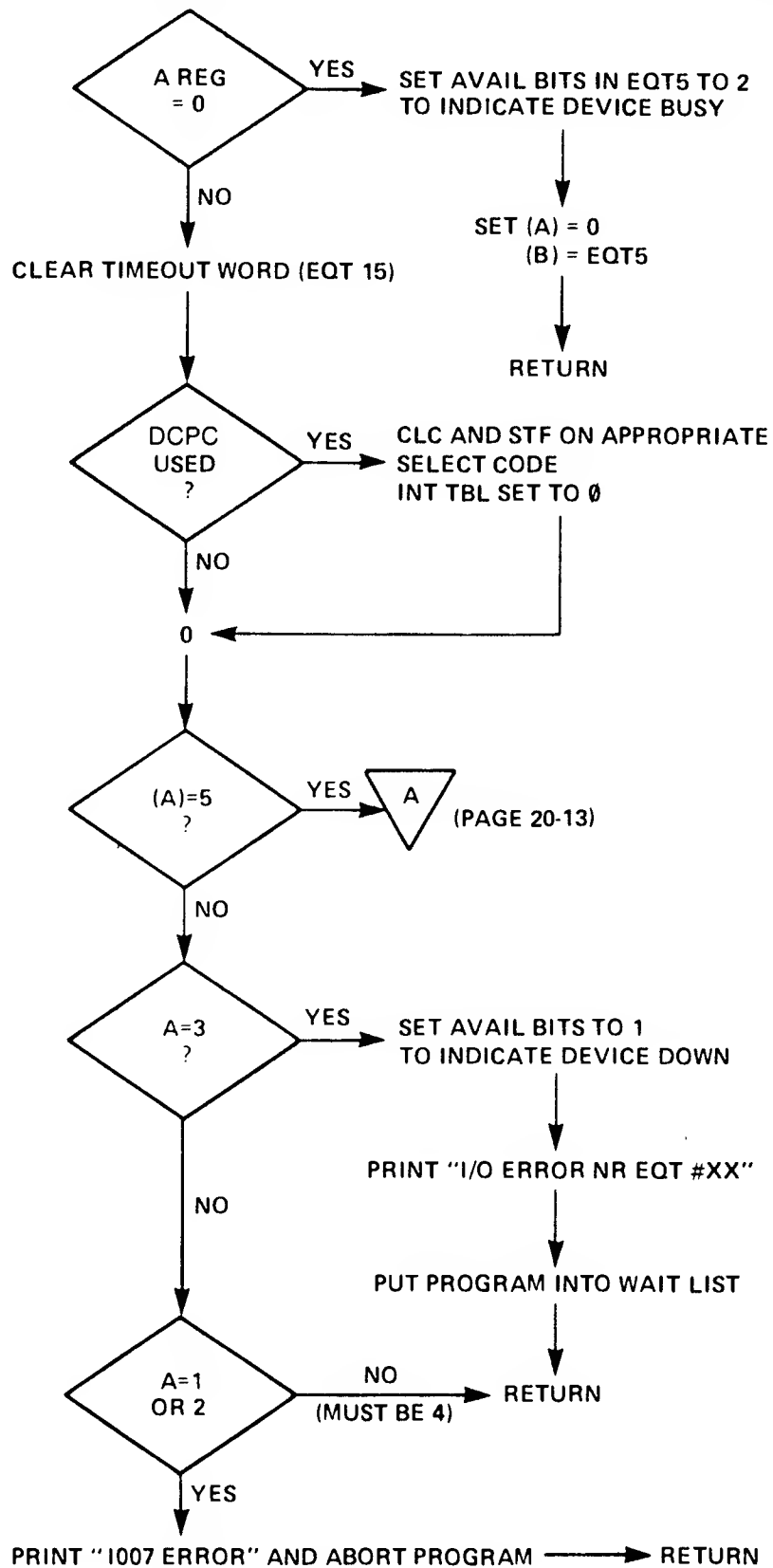


COMMUNITY CHEST  
RETURN TO JAIL.  
DO NOT PASS GO.  
DO NOT COLLECT \$200.

SET THE A-REGISTER TO INDICATE INITIATION OR  
REJECTION

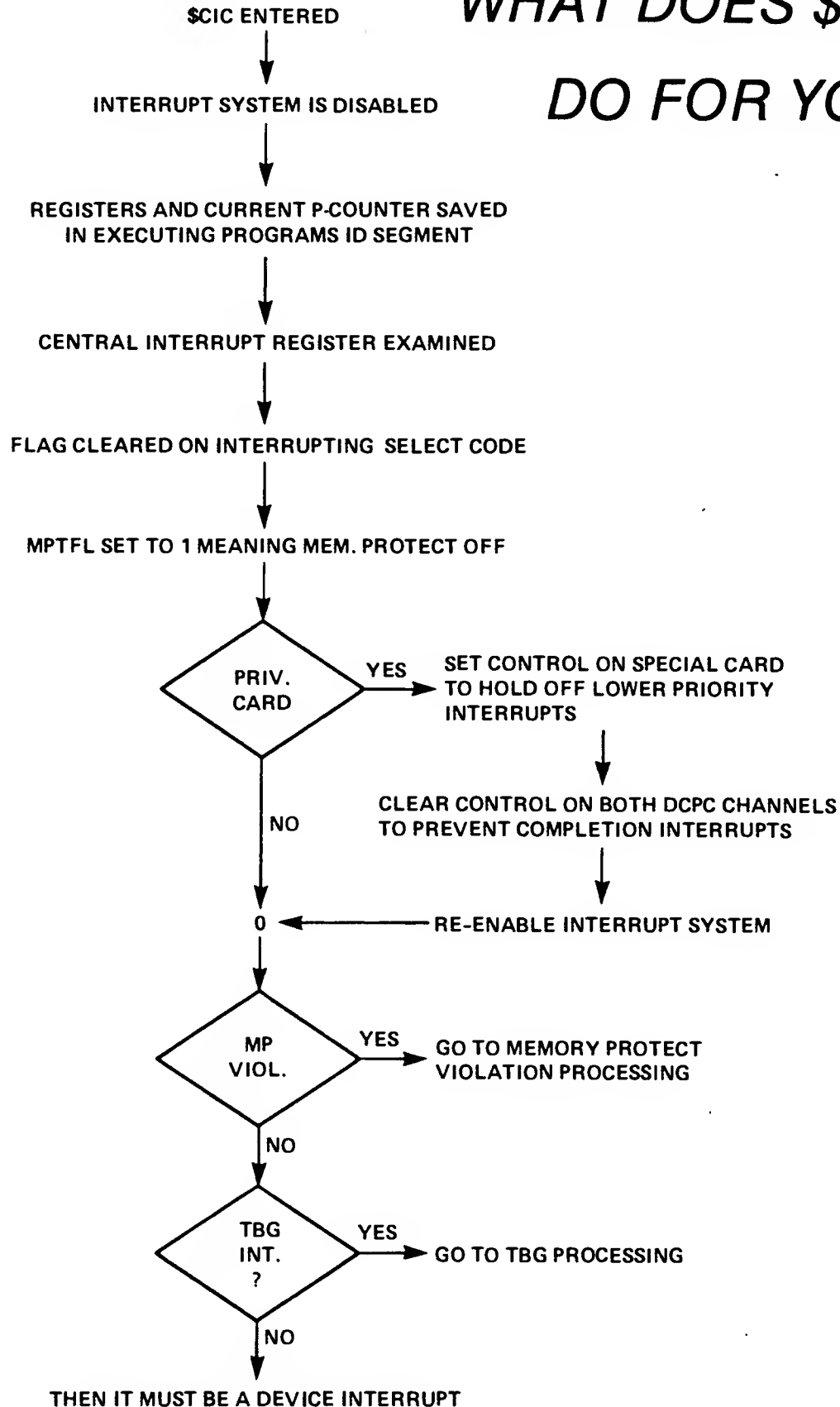
A = 0	OPERATION INITIATED
= 1	READ OR WRITE ILLEGAL
= 2	CONTROL REQUEST ILLEGAL
= 3	EQUIPMENT NOT READY
= 4	IMMEDIATE COMPLETION
= 5	DCPC CHANNEL REQUIRED
= 6-99	PROGRAM MAKING I/O REQUEST IS ABORTED & MESSAGE PRINTED
6-59	HP DRIVERS
60-99	USER WRITTEN DRIVERS

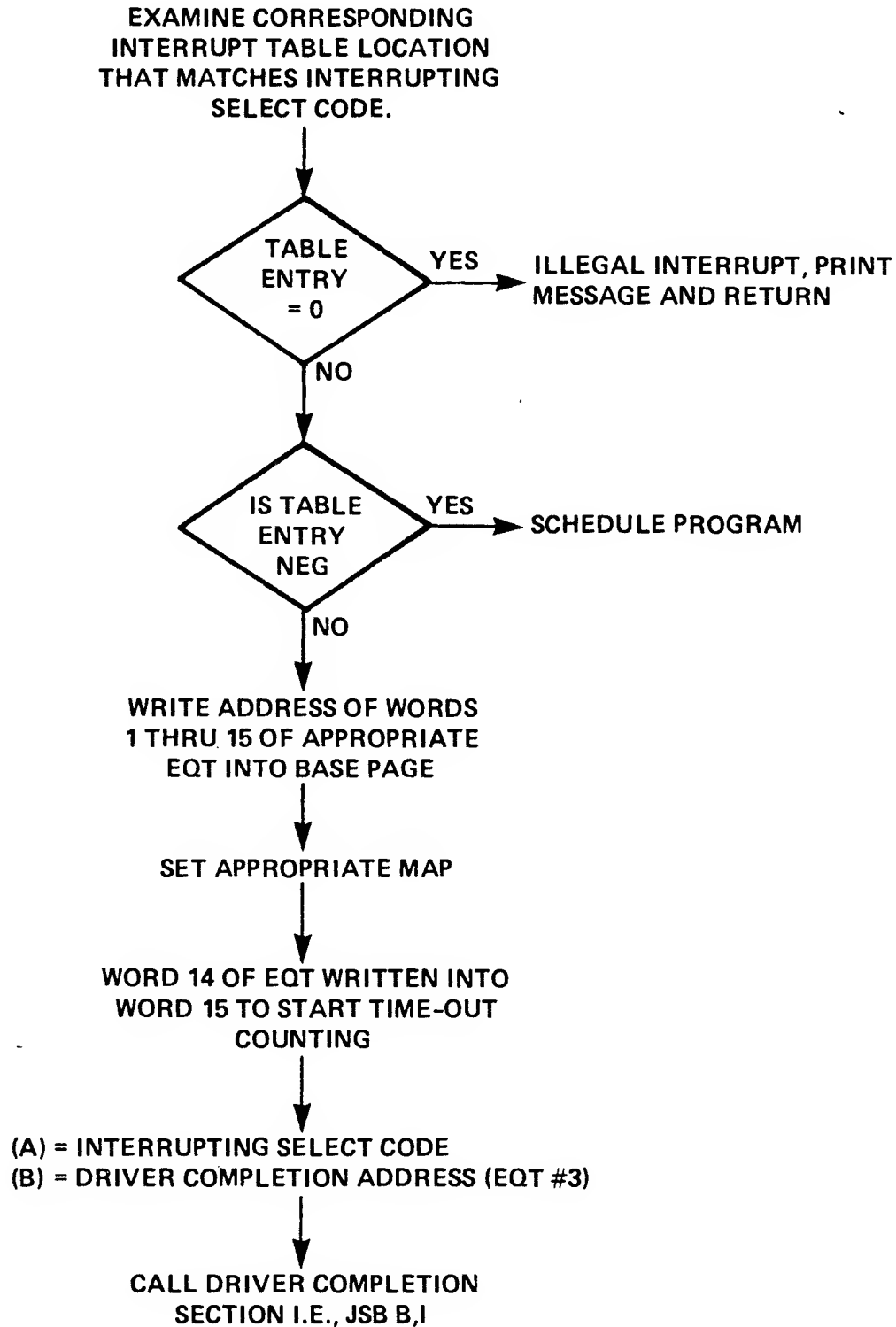
## ON RETURN FROM YOUR DRIVER, WHAT HAPPENS?





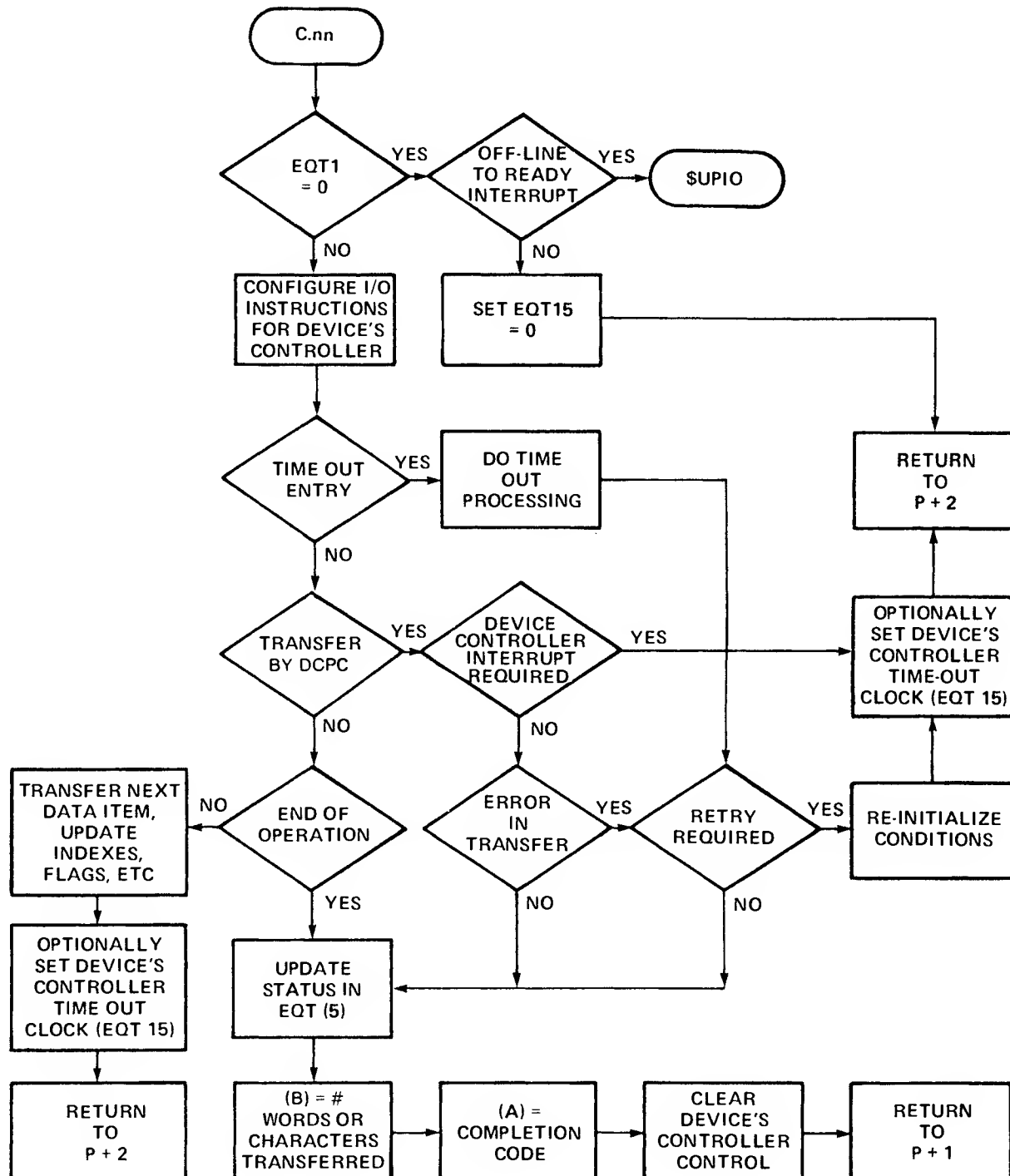
# WHAT DOES \$CIC DO FOR YOU?





**WHAT DOES \$CIC DO FOR YOU (CONT'D)**

# RTE DRIVER COMPLETION SECTION



RTE DRIVERS MUST INDICATE TO THE  
EXECUTIVE WHEN TRANSFER IS COMPLETE.  
THIS IS ACCOMPLISHED BY EXITING THE DRIVER  
IN ONE OF TWO WAYS

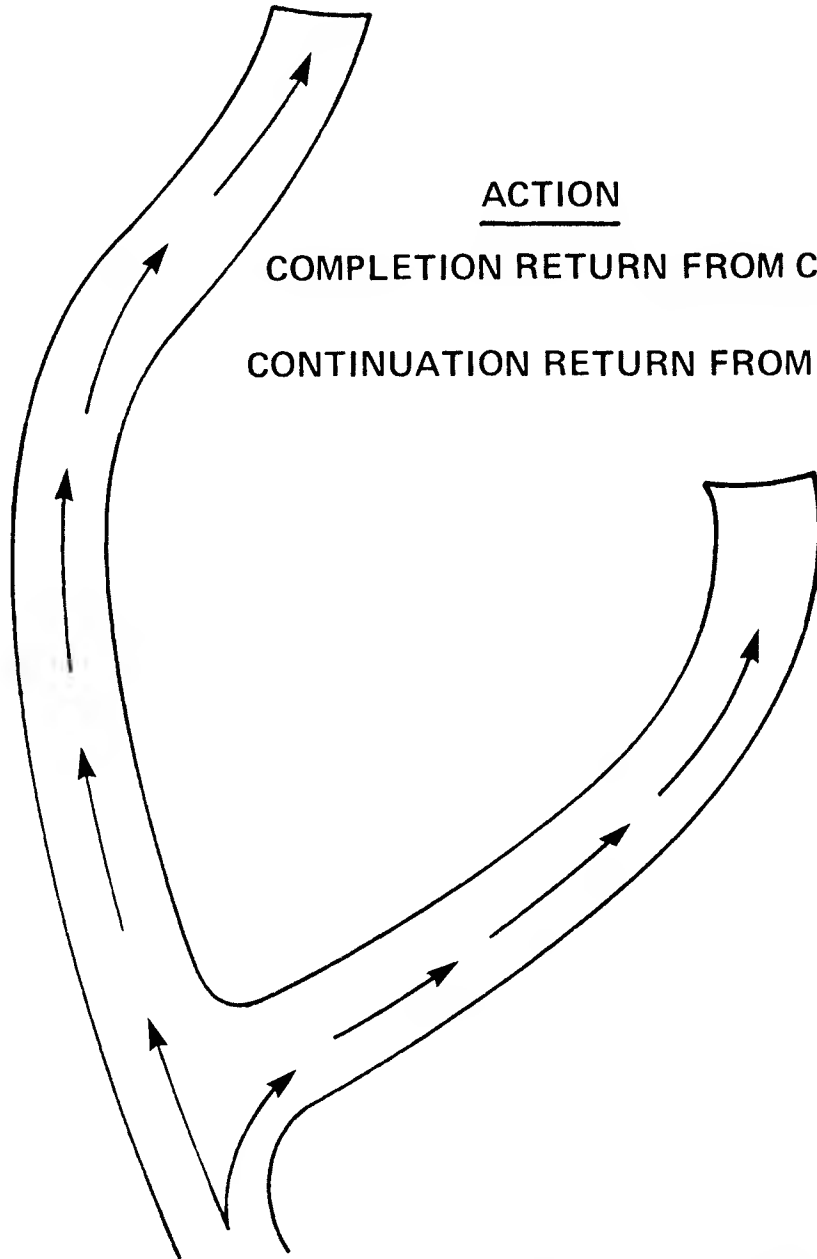
DRIVER  
RETURN  
TO RTE

- I.  $P + 1$
- II.  $P + 2$

ACTION

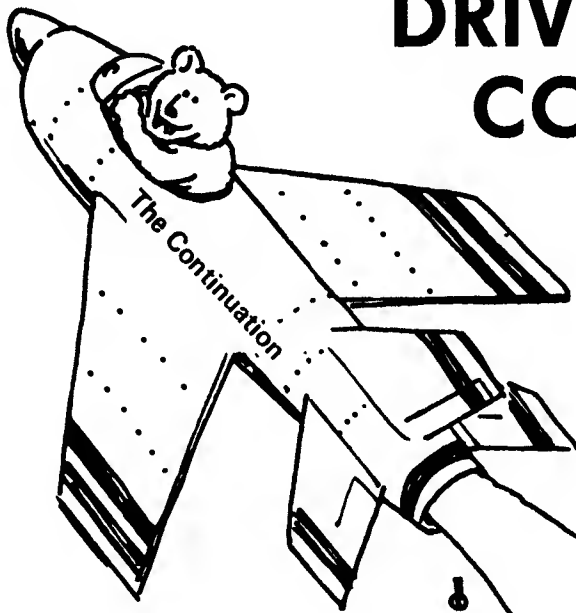
COMPLETION RETURN FROM C.nn

CONTINUATION RETURN FROM C.nn



**TWO WAYS TO RETURN FROM  
THE COMPLETION SECTION**

# DRIVER RULES FOR CONTINUATION RETURN



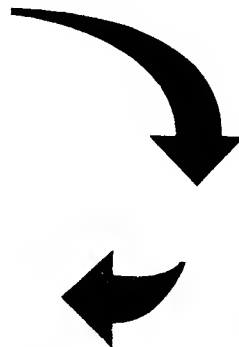
- RESTART DEVICE
- INCREMENT C.nn
- RETURN THROUGH C.nn
- CONTENTS OF A AND B  
REGISTERS MEANINGLESS

## DRIVER RULES FOR COMPLETION RETURN

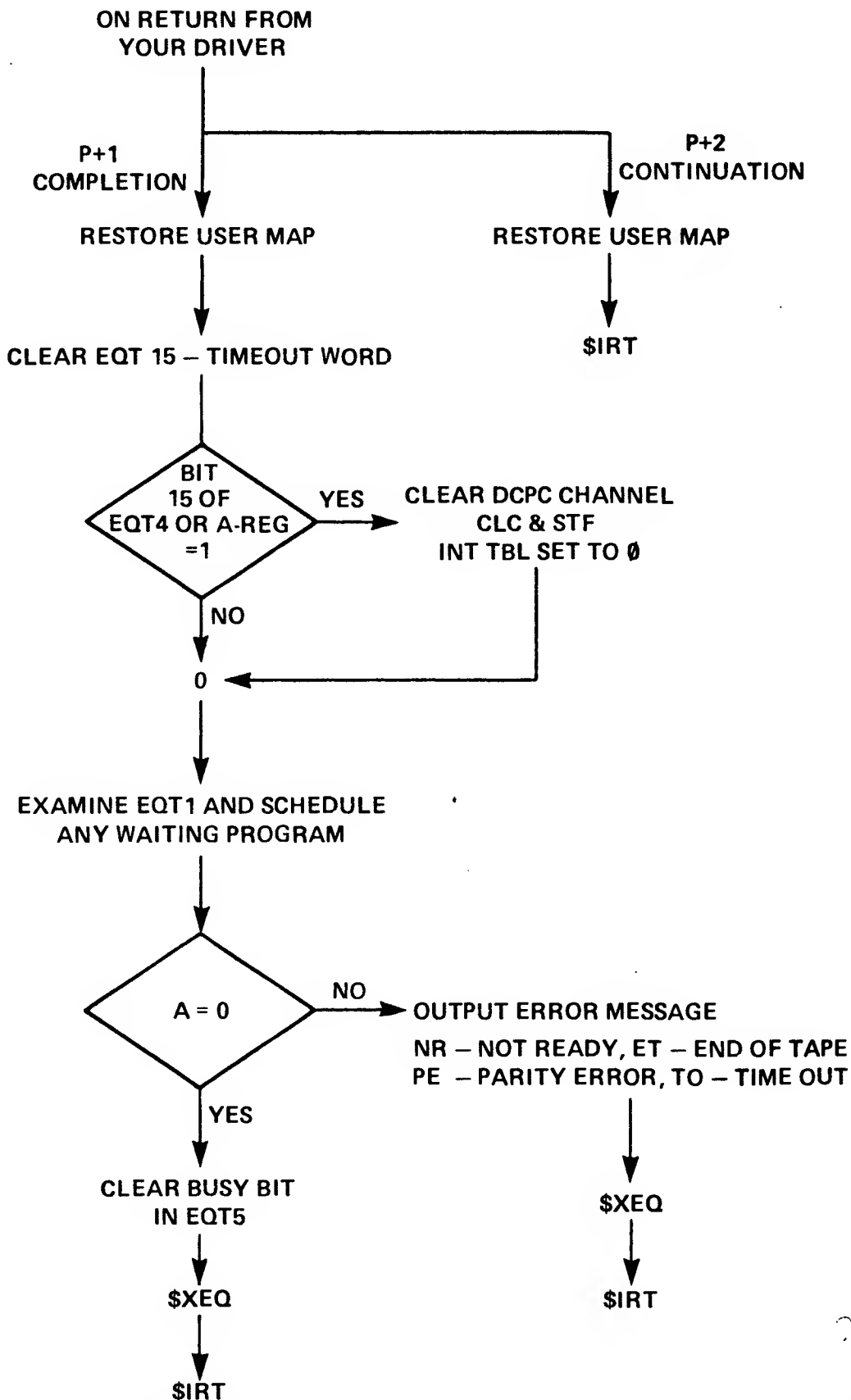
- CLC ON DEVICE
- PUT COMPLETION STATUS IN A-REGISTER
  - (A) = 0 FOR SUCCESSFUL COMPLETION
  - = 1 DEVICE NOT READY
  - = 2 END OF TRANSMISSION (UNEXPECTED)
  - = 3 TRANSMISSION PARITY ERROR
  - = 4 DEVICE TIME-OUT
- NOTE: (A) #0 PRODUCES AN ERROR MESSAGE ON THE SYSTEM CONSOLE.

<u>STATUS IN A</u>	<u>ERROR MESSAGE</u>
1	I/O NR En Ln Sn
2	I/O ET En Ln Sn
3	I/O PE En Ln Sn
4	I/O TO En Ln Sn

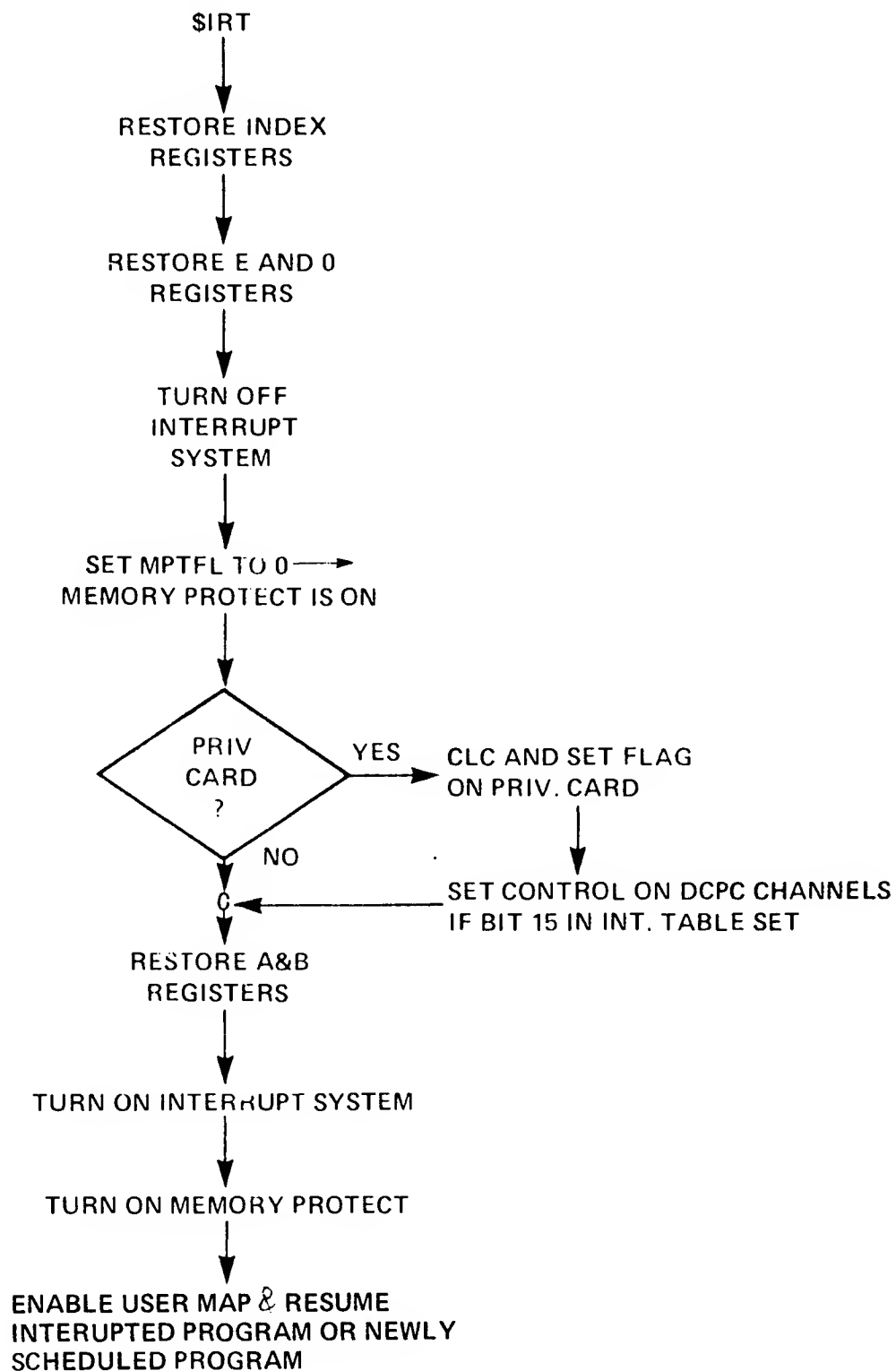
- PUT THE TRANSMISSION LOG IN THE B-REGISTER
- MODIFY STATUS BITS 0 THROUGH 7 OF EQT WORD 5.



# WHAT DOES \$CIC DO AFTER YOUR DRIVER FINISHES?



# \$CIC RETURN SEQUENCE





\*\* STANDARD RTE DRIVER EXAMPLE \*\*

```

0001          ASMB,L
0003 00000      NAM DVR70  ** STANDARD RTE DRIVER EXAMPLE **
0004*
0005*
0006          ENT I.70,C.70
0007*
0008*
0009* DRIVER 70 OPERATES UNDER THE CONTROL OF THE I/O CONTROL (IOC)
0010* AND THE CENTRAL INTERRUPT CONTROL (CIC) MODULES OF RTE.
0011* THIS DRIVER IS RESPONSIBLE FOR CONTROLLING OUTPUT
0012* TRANSMISSION TO A 16 BIT EXTERNAL DEVICE.
0013* I.70 IS THE ENTRY POINT FOR THE *INITIATION* SECTION
0014* AND C.70 IS THE ENTRY POINT FOR THE *CONTINUATION/COMPLETION*
0015* SECTION.
0016*
0017* NOTE THAT THIS DRIVER DOES NOT PROCESS TIME-OUTS OR
0018* POWER FAIL. THESE PROCEDURES ARE LEFT ENTIRELY UP TO
0019* THE SYSTEM.
0020*
0021* REMEMBER THAT RTE SETS THE ADDRESSES OF EACH WORD OF
0022* THE 15 WORD EQT ENTRY FOR THE DEVICE BEING SERVICED INTO
0023* THE BASE PAGE COMMUNICATIONS AREA ON EACH ENTRY TO THE
0024* DRIVER.
0025* THIS DRIVER REFERENCES THESE ADDRESSES THROUGH VARIABLES
0026* EQT1 THROUGH EQT15.
0027*
0028* *****
0029* * INITIATION SECTION *
0030* *****
0031*
0032* THE INITIATION SECTION IS CALLED FROM I/O CONTROL (IOC) TO
0033* INITIALIZE A DEVICE AND INITIATE AN OUTPUT OPERATION
0034*
0035* THE CALLING SEQUENCE FOR THE INITIATION SECTION IS:
0036*
0037*          (SET A = SELECT CODE OF I/O DEVICE)
0038* P      JSB I.70
0039* P+1    (RETURN POINT)
0040*
0041*      ON RETURN, A REGISTER INDICATES STATUS, AS FOLLOWS:
0042*
0043*          A = 0,  OPERATION SUCCESSFULLY INITIATED
0044*          A NOT 0, OPERATION REJECTED FOR THE FOLLOWING
0045*                  REASON:
0046*
0047*                  A = 1 = ILLEGAL READ REQUEST
0048*                  A = 2 = ILLEGAL CONTROL REQUEST
0049*
0050* (NOTE, HOWEVER, THAT A "CLEAR" CONTROL REQUEST FROM THE
0051* SYSTEM WILL BE PROCESSED BY THE DRIVER, AS REQUIRED.)
0052*
0053* *****
0054* * CONTINUATION/COMPLETION SECTION *
0055* *****
0056*
0057* THE CONTINUATION/COMPLETION SECTION IS CALLED BY CENTRAL

```

★ ★ STANDARD RTE DRIVER EXAMPLE ★ ★

0058\* INTERRUPT CONTROL (CIC) TO CONTINUE OR COMPLETE AN OPERATION WHEN  
0059\* AN INTERRUPT IS DETECTED ON THE DEVICE

0060\*

0061\* THE CALLING SEQUENCE FOR THE COMPLETION SECTION IS:

0062\*

0063\*

(SET A = SELECT CODE OF I/O DEVICE)

0064\*

P JSB C.70

0065\*

P+1 COMPLETION RETURN

0066\*

P+2 CONTINUATION RETURN

0067\*

0068\*

ON RETURN, A & B REGISTERS INDICATE STATUS, AS FOLLOWS:

0069\*

0070\*

ON A COMPLETION RETURN:

0071\*

0072\*

A = 0, SUCCESSFUL COMPLETION, WITH

0073\*

B = NUMBER OF WORDS TRANSMITTED

0074\*

0075\*

A = 2, TRANSMISSION ERROR DETECTED

0076\*

0077\*

ON A CONTINUATION RETURN, THE REGISTERS ARE

0078\*

MEANINGLESS

0079\*

0080\* RECORD FORMAT:

0081\*

0082\*

THIS DRIVER PROVIDES A 16 BIT BINARY WORD

0083\*

TRANSFER ONLY.

0084\*

★ ★ STANDARD DRIVER - INITIATION SECTION ★ ★

```

0086*
0087*
0088*
0089*
0090*
0091 00000 000000 I.70 NOP ENTRY FROM IOC
0092*
0093 00001 016100R JSB SETIO CONFIGURE I/O INSTRUCTIONS FOR DEVICE
0094*
0095 00002 161665 LDA EQT6,I GET CONTROL WORD OF REQUEST, AND
0096 00003 012115R AND =B3 ISOLATE THE REQUEST TYPE
0097*
0098 00004 052116R CPA =B1 IF REQUEST IS FOR INPUT
0099 00005 126000R JMP I.70,I THEN REJECT IT (A = 1 = ILLEGAL READ)
0100 00006 052117R CPA =B2 IF REQUEST IS FOR OUTPUT
0101 00007 026017R JMP D.X1 THEN GO PROCESS WRITE REQUEST
0102*
0103* CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
0104* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
0105*
0106 00010 161665 LDA EQT6,I ACCESS CONTROL WORD
0107 00011 012120R AND =B3700 ISOLATE SUBFUNCTION
0108 00012 002002 SZA "CLEAR" REQUEST?
0109 00013 026015R JMP REJCT NO, SO REJECT REQUEST AS ILLEGAL
0110*
0111 00014 106700 I.0 CLC SC YES, CLEAR DEVICE AND RETURN
0112*
0113* REQUEST ERROR - CAUSE REJECT RETURN TO IOC
0114*
0115 00015 052117R REJCT LDA =B2 SET A = 2 FOR ILLEGAL CONTROL REQUEST
0116 00016 126000R JMP I.70,I AND RETURN (A = 2 = ILLEGAL CONT. REQ.)
0117*
0118* WRITE REQUEST PROCESSING
0119*
0120 00017 161660 D.X1 LDA EQT7,I GET REQUEST BUFFER ADDRESS
0121 00020 171670 STA EQT9,I AND SET IT AS CURRENT ADDRESS
0122 00021 161667 LDA EQT8,I GET REQUEST BUFFER LENGTH
0123 00022 003004 CMA,INA MAKE NEGATIVE AND
0124 00023 171671 STA EQT10,I AND SAVE AS REMAINING BUFFER LENGTH
0125 00024 002002 SZA IS BUFFER LENGTH = 0?
0126 00025 026031R JMP D.X3 NO, PROCESS AS USUAL
0127 00026 052121R LDA =B4 YES, SO MAKE IMMEDIATE COMPLETION RETURN
0128 00027 006400 CLR SET TRANSMISSION LOG = 0 INTO B
0129 00030 126000R JMP I.70,I AND RETURN (A = 4 = IMMED. COMPLETION)
0130*
0131* CALL THE CONTINUATION/COMPLETION SECTION TO WRITE FIRST WORD
0132*
0133 00031 052114R D.X3 LDA P2 ADJUST RETURN ADDRESS SO WILL
0134 00032 072036R STA C.70 RETURN HERE (INITIATION SECTION)
0135 00033 026047R JMP D.X2 GO TO COMPLETION SECTION
0136*
0137 00034 002400 IEXIT CLA NOW RETURN TO IOC WITH
0138 00035 126000R JMP I.70,I OPERATION INITIATED (A = 0 = OK)
139*

```

\*\* STANDARD DRIVER - CONTINUATION/COMPLETION SECTION \*\*

```

0141*
0142*
0143* *****
0144* * CONTINUATION/COMPLETION SECTION *
0145* *****
0146 00036 000000 C.70 NOP CONTINUATION/COMPLETION ENTRY POINT
0147*
0148 00037 016100R JSB SETIO CONFIGURE I/O INSTRUCTIONS
0149*
0150 00040 161660 LDA EQT1,I CHECK FOR SPURIOUS INTERRUPT
0151 00041 012122R AND =B77777 ISOLATE I/O REQUEST LIST PTR (15 BITS)
0152 00042 002002 SZA IS A REQUEST IN PROGRESS?
0153 00043 026047R JMP D.X2 YES, GO PROCESS REQUEST
0154*
0155 00044 171774 STA EQT15,I NO, SPURIOUS INTERRUPT=ZERO TIME-OUT CLK
0156 00045 036036R ISZ C.70 ADJUST RETURN TO P+2 (CONTINUATION)
0157 00046 126036R JMP C.70,1 MAKE CONTINUATION RETURN TO CIC
0158*
0159 00047 002400 D.X2 CLA IF CURRENT BUFFER LENGTH = 0,
0160 00050 151071 CPA EQT10,I THEN GO TO STATUS
0161 00051 026063R JMP I.3 SECTION. (I.E., TRANSFER DONE NOW)
0162*
0163 00052 165670 LDB EQT9,I GET CURRENT BUFFER ADDRESS
0164*
0165 00053 135670 ISZ EQT9,1 ADD 1 FOR NEXT WORD
0166 00054 160001 LDA B,I GET WORD TO BE WRITTEN TO DEVICE
0167 00055 135671 ISZ EQT10,I INCREMENT WORD COUNT ALSO
0168 00056 000000 NOP IGNORE P+1 SKIP IF LAST WORD
0169*
0170 00057 102000 I.1 STA SC OUTPUT WORD TO INTERFACE
0171 00060 100700 I.2 STC SC,C TURN DEVICE ON
0172*
0173 00061 006136R ISZ C.70 ADJUST RETURN TO P+2 (CONTINUATION)
0174 00062 126036R JMP L.70,1 MAKE CONTINUATION RETURN
0175*
0176* STATUS AND COMPLETION SECTION
0177*
0178 00063 102500 I.3 LIA SC GET STATUS WORD FROM DEVICE
0179 00064 012120R AND =B77 STRIP OFF UNUSED BITS
0180 00065 070001 STA B SAVE IN B TEMPORARILY
0181 00066 101004 LDA EQT5,I REMOVE PREVIOUS STATUS
0182 00067 012124R AND =B177400 BITS IN EQT WORD 5
0183 00070 030001 LOR B OR IN NEW BITS
0184 00071 171064 STA EQT5,I AND RESET INTO EQT WORD 5
0185*
0186 00072 102400 CLA SET A = 0 = OK RETURN CODE
0187 00073 056121R CPB =B4 ERROR STATUS BIT ON?
0188 00074 002117R LDA =B2 YES, SET A = 2 = ERROR RETURN
0189*
0190 00075 165667 LDB EQT8,I SET B = TRANSMISSION LOG
0191*
0192 00076 106700 I.4 CLC SC CLEAR DEVICE CONTROLLER
0193*
0194 00077 126036R JMP C.70,1 MAKE COMPLETION RETURN TO CIC
0195*

```

★★ STANDARD DRIVER - SUBROUTINE SETIO ★★

```

0197*
0198*
0199*          *****
0200*          * SUBROUTINE SETIO *
0201*          *****
0202* SUBROUTINE <SETIO> CONFIGURES ALL I/O INSTRUCTIONS IN DRIVER
0203*
0204* 00100 000000 SETIO NOP          ENTRY POINT
0205*
0206* 00101 032113R      IOR LIA      COMBINE LIA WITH I/O
0207* 00102 072063R      STA I,3      SELECT CODE AND SET IN CODE
0208*
0209* 00103 042125R      ADA #B100     CONSTRUCT OTA INSTRUCTION
0210* 00104 072057R      STA I,1
0211*
0212* 00105 042126R      ADA #B1100    CONSTRUCT STC,C INSTRUCTION
0213* 00106 072060R      STA I,2
0214*
0215* 00107 032127R      IOR #B4000    CONSTRUCT CLC INSTRUCTION
0216* 00110 072014R      STA I,0
0217* 00111 072076R      STA I,4
0218*
0219* 00112 126100R      JMP SETIO,I    RETURN
0220*

```

\*\*\* STANDARD DRIVER - DATA AREA \*\*\*

```

0222*
0223*          *****
0224*          * DATA AREA *
0225*          *****
0226*
0227* CONSTANT AND STORAGE AREA
0228*
0229 000000      A      EQU 0      A-REGISTER
0230 000001      B      EQU 1      B-REGISTER
0231*
0232 000000      SC      EQU 0      DUMMY I/O SELECT CODE NUMBER
0233 00113 102000 LIA      LIA 0      CODE FOR LIA INSTRUCTION
0234 00114 000033R P2      DEF IEXIT-1  RETURN POINT IN INITIATION SECTION
0235*
0236* ** BASE PAGE COMMUNICATIONS AREA DEFINITIONS **
0237*
0238 01650      .      EQU 1650B
0239*
0240 01660      EQT1 EQU .+8
0241 01661      EQT2 EQU .+9
0242 01662      EQT3 EQU .+10
0243 01663      EQT4 EQU .+11
0244 01664      EQT5 EQU .+12
0245 01665      EQT6 EQU .+13
0246 01666      EQT7 EQU .+14
0247 01667      EQT8 EQU .+15
0248 01670      EQT9 EQU .+16
0249 01671      EQT10 EQU .+17
0250 01672      EQT11 EQU .+18
0251 01771      EQT12 EQU .+81
0252 01772      EQT13 EQU .+82
0253 01773      EQT14 EQU .+83
0254 01774      EQT15 EQU .+84
0255*
0256*
00115 0000003
00116 0000001
00117 0000002
00120 003700
00121 0000004
00122 077777
00123 0000077
00124 177400
00125 000100
00126 001110
00127 004000
0257          END
** NO ERRORS *TOTAL ***ATE ASMB 700924**

```

**PURPOSE — DCPC TRANSFERS DATA DIRECTLY  
BETWEEN MEMORY AND HIGH  
SPEED AND/OR SYNCHRONOUS  
DEVICES.**

# **DCPC DUAL CHANNEL PORT CONTROLLER**

**THE TRANSFER IS BEGUN BY THE INITIATOR  
PORTION OF DRIVER. OPERATION IS  
CONTROLLED BY COMPUTER HARDWARE.**

# **FUNCTION OF THE DCPC INITIATOR**

**\* SETS UP DCPC HARDWARE**

**\* SPECIFIES DIRECTION OF TRANSFER  
(TO OR FROM MEMORY)**

**\* SPECIFIES WHERE IN MEMORY TO READ OR STORE DATA**

**\* DECIDES WHICH DEVICE SELECT CODE IS TO BE USED**

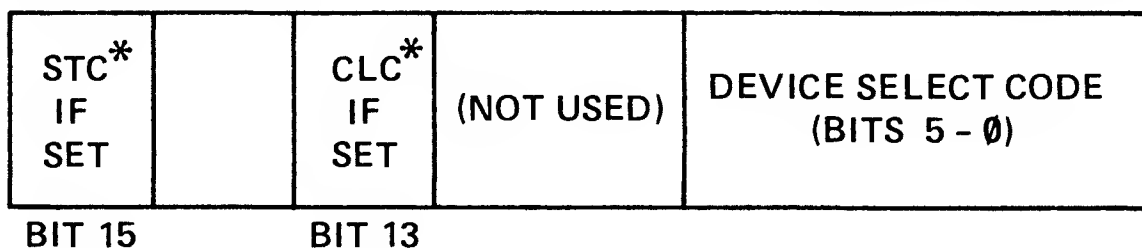
**\* CONTROLS HOW MUCH DATA IS TO BE TRANSFERRED**



# DCPC CONTROL WORDS

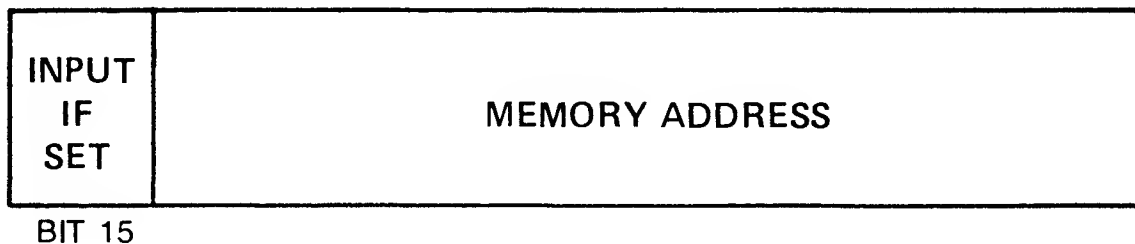
## CONTROL WORD 1

(DEVICE CONTROL)



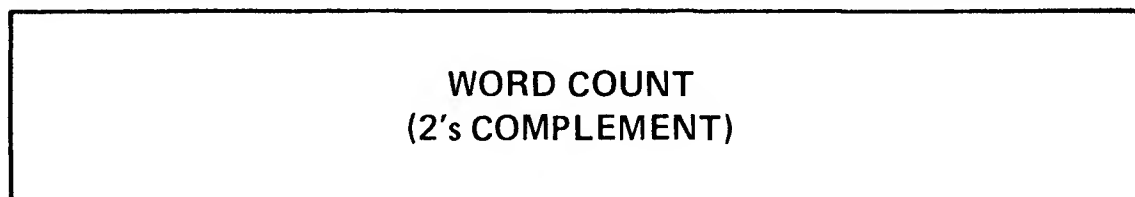
## CONTROL WORD 2

(MEMORY CONTROL)



## CONTROL WORD 3

(BLOCK LENGTH CONTROL)



\*STC } IF SET IN CONTROL WORD 1, DCPC WILL ISSUE  
CLC } STC FOR EACH WORD TRANSFER AND/OR CLC  
UPON COMPLETION.

# DCPC USAGE IN RTE DRIVERS

A DRIVER OBTAINS A DCPC CHANNEL FOR USE BY:

♦ DCPC BIT SET IN EQT AT SYSTEM GENERATION TIME

OR

♦ THE DRIVER CAN DYNAMICALLY REQUEST A DCPC CHANNEL IN THE INITIATOR.

A REG = 5 ON RETURN

## DCPC ASSIGNMENT BY RTE

Before calling the driver initiation section:

- CHAN is setup on the base page
- Words 1 and 2 of the interrupt table are setup where,

DCPC Channel 1 Assignment Word	Interrupt Table Word 1 (I/O SELECT CODE 6)
DCPC Channel 2 Assignment Word	Interrupt Table Word 2 (I/O SELECT CODE 7)

Where each DCPC Channel Assignment Word has the format:

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<div style="display: flex; justify-content: space-between;"> <span>F!</span> <span>Address</span> </div>

Where:

- F           = 1, if the driver assigned to the channel needs the DCPC completion interrupt (set only in systems with a privileged interrupt card).
- = 0, otherwise
- Address     = the address of the EQT entry of the driver to which the DCPC channel is assigned.
- = 0, if the DCPC channel is currently not assigned.

## DYNAMIC DCPC REQUEST

- Once the driver determines that it needs a DCPC channel for a request, it requests DCPC by:

```
CHDCP EQU *           Executes this code if DCPC required
DLD INTBA,I           Access DCPC Channel Assignment Words
CPA EQT1              Is DCPC channel 1 assigned to this driver?
JMP CH1               Yes, configure and initiate transfer on channel 1
CPB EQT1              Is DCPC channel 2 assigned to this driver?
JMP CH2               Yes, configure and initiate transfer on channel 2
LDA =B5               No. A DCPC channel is not assigned. Set
JMP IXnn,I            A = 5 to request one from IOC, and return.
```

- When the request is completed, the DCPC channel is returned by:

```
LDA COMCD             Set A = completion code determined earlier
IOR =B100000          Set sign bit to indicate dynamic DCPC assignment
JMP CXnn,I            Return to CIC
```

# DCPC COMPLETION INTERRUPT

- BOTH THE DCPC AND DEVICE CAN GENERATE INTERRUPTS ON COMPLETION
- IF YOUR DRIVER NEEDS ONLY A DEVICE INTERRUPT, CLEAR CONTROL ON THE DCPC CHANNEL AFTER INITIALIZATION

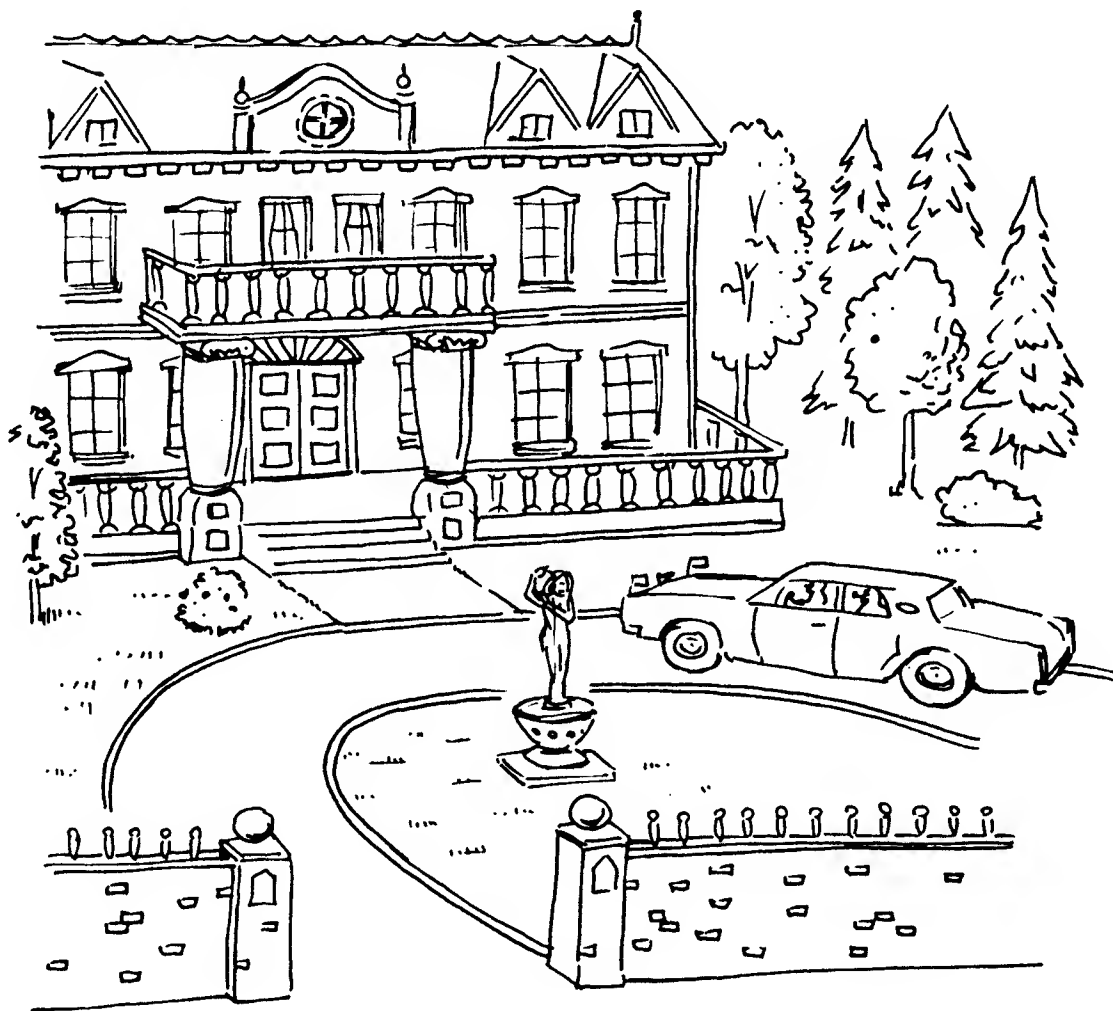
NO FURTHER PROCESSING IS REQUIRED

- IF A DCPC COMPLETION INTERRUPT IS REQUIRED, THEN SPECIAL PROCESSING IS REQUIRED IN YOUR DRIVER

SPECIAL PROCESSING IF DCPC  
COMPLETION INTERRUPT IS  
REQUIRED

CLF 0	Disable the interrupt system
STC DCPC,C	Initiate transfer on DCPC channel
CLA	Bypass section below if
CPA DUMMY	DUMMY = 0(non-privileged system)
JMP X	and special processing not needed.
CLC DCPC	Clear DCPC control to inhibit DCPC
LDB INTBA	interrupt. Set B = address of the appropriate
LDA CHAN	DCPC Channel Assignment word in the
CPA = D7	Interrupt Table
INB	
LDA B,I	Set bit 15 of DCPC channel assignment entry
IOR = B1000000	equal to 1 as flag to system to turn DCPC
STA B,I	interrupts back on later. Reenable the
STF 0	interrupt system.
X   EQT *	Continue processing.

# PRIVILEGED DRIVERS



# **WHAT IS A PRIVILEGED INTERRUPT?**

- \* RTE NORMALLY HAS THE INTERRUPT SYSTEM OFF WHILE A DRIVER SERVICES AN I/O REQUEST.**
- \* SOME DEVICES CANNOT BE DELAYED AND SHOULD HAVE THE "PRIVILEGE" OF GENERATING AN INTERRUPT AT ANY TIME.**
- \* THIS REQUIRES A I/O CARD KNOWN AS A PRIVILEGED FENCE.**
- \* THIS FENCE PHYSICALLY SEPARATES THE PRIVILEGED DEVICE INTERRUPTS FROM REGULAR DEVICE INTERRUPTS.**
- \* THE SELECT CODE OF FENCE USED IS STORED IN BASE PAGE LOCATION 1737<sub>8</sub> LABELED "DUMMY".**
- \* RTE OPERATES WITH AN INTERRUPT SYSTEM ON FOR DRIVER SERVICING BUT INTERRUPTS ARE HELD OFF FOR THOSE DEVICES AFTER THE FENCE.**

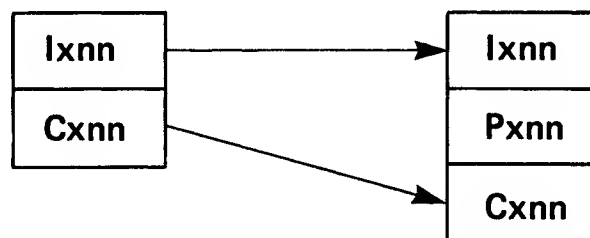


## HOW ARE PRIVILEGED INTERRUPTS PROCESSED?

- PRIVILEGED DRIVER
  - USER PROGRAM CALLS ARE THE SAME AS ANY I/O CALL
  - THE DRIVER, IN GENERAL, HAS THE SAME STRUCTURE AS A REGULAR DRIVER PLUS A PRIVILEGED PORTION.
- PRIVILEGED ROUTINE
  - TRAP CELL SET TO JSB XXX,I DURING SYSGEN.

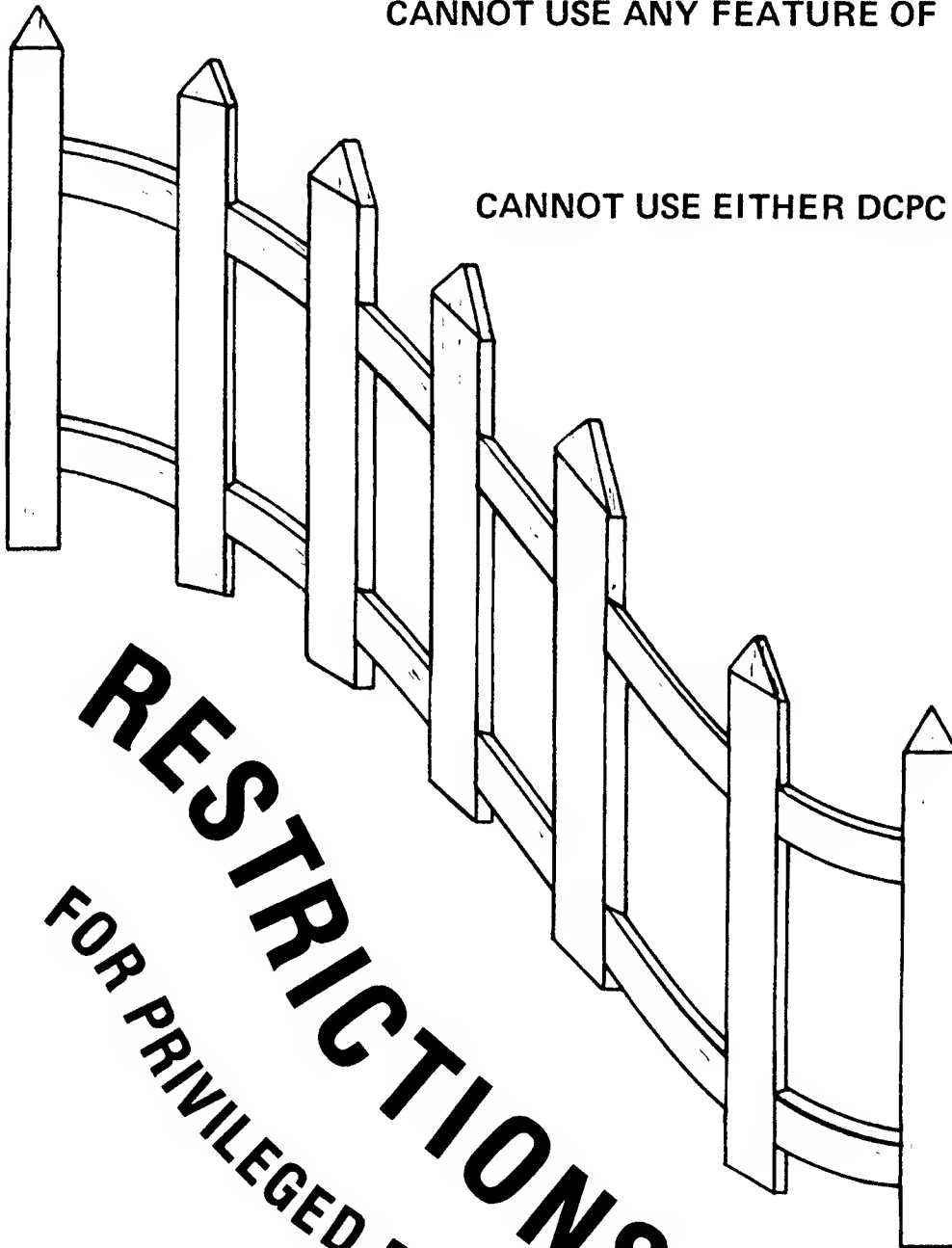
STANDARD  
DRIVER

PRIVILEGED  
DRIVER



CANNOT USE ANY FEATURE OF **EXEC**

CANNOT USE EITHER DCPC CHANNEL.



**RESTRICTIONS**  
FOR PRIVILEGED ROUTINES

# **DATA TRANSFER**

- USE SYSTEM COMMON

- CLASS I/O –  
SYSTEM AVAILABLE MEMORY

- REIO – SYSTEM AVAILABLE MEMORY

- IF THESE ARE USED, THEN YOUR DRIVER NEVER  
NEEDS TO USE MAP SWITCHING IN RTE-IV

# PRIVILEGED DRIVER CONCEPTS

- CALLED BY EXEC OR REIO I/O CALL
- CALLING PROGRAM PLACED INTO I/O SUSPENSION
- DEVICE TRAP CELL CHANGED FROM

**JSB \$CIC,I** TO **JSB P.XX,I**

NOW RTE BYPASSED ON INTERRUPTS.

- SYSTEM IS NOTIFIED OF COMPLETION BY:

PRIVILEGED PORTION OF DRIVER SETS  
TIMEOUT IN EQT AND EXITS.  
ON TIMEOUT, RTE ENTERS CONTINUATOR

- **C.XX** RETURNS TRANSMISSION LOG AND STATUS
- SUSPENDED PROGRAM RESUMES

- CHECKS FOR VALID REQUEST CODE
- SINCE DRIVER CONTROLS ONLY ONE DEVICE, CONFIGURE ONCE AND SET A SWITCH TO PREVENT RE-EXECUTING.
- TRAP CELL MODIFIED ONCE
- COUNT AND BUFFER ADDRESS SAVED WITHIN THE DRIVER
- START DEVICE
- RETURN

## INITIATOR SECTION

# PRIVILEGED SECTION

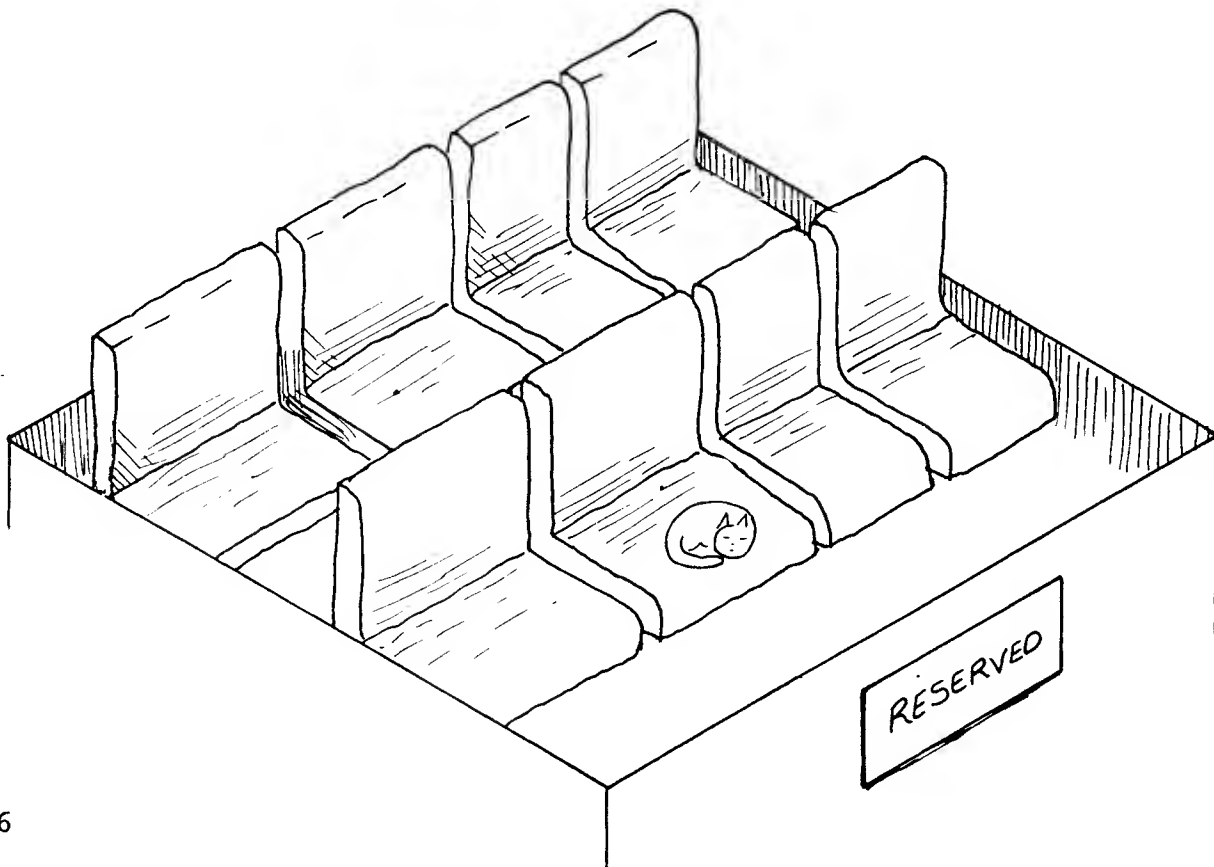
- ENTRY IS AUTOMATIC, BYPASSING RTE

## TURN OFF INTERRUPT SYSTEM

- SAVE ALL REGISTERS TO BE USED
- DISABLE DCPC INTERRUPTS
- SAVE MEMORY PROTECT STATUS
- SAVE DMS STATUS

## ENABLE INTERRUPT SYSTEM

- TRANSFER DATA



# **MORE DATA TO BE PROCESSED**

- **DISABLE INTERRUPT SYSTEM**

- **START DEVICE**

- **REENABLE DCPC COMPLETION INTERRUPT IF:**

**MP WAS ON AND A STANDARD DRIVER REQUIRES  
A DCPC COMPLETION INTERRUPT.**

- **RESTORE MEMORY PROTECT AND ITS FLAG  
(MPTFL) TO ITS ORIGINAL STATE**

- **RESTORE REGISTERS**

- **ENABLE INTERRUPT SYSTEM**

- **RESTORE DMS**

- **RETURN**

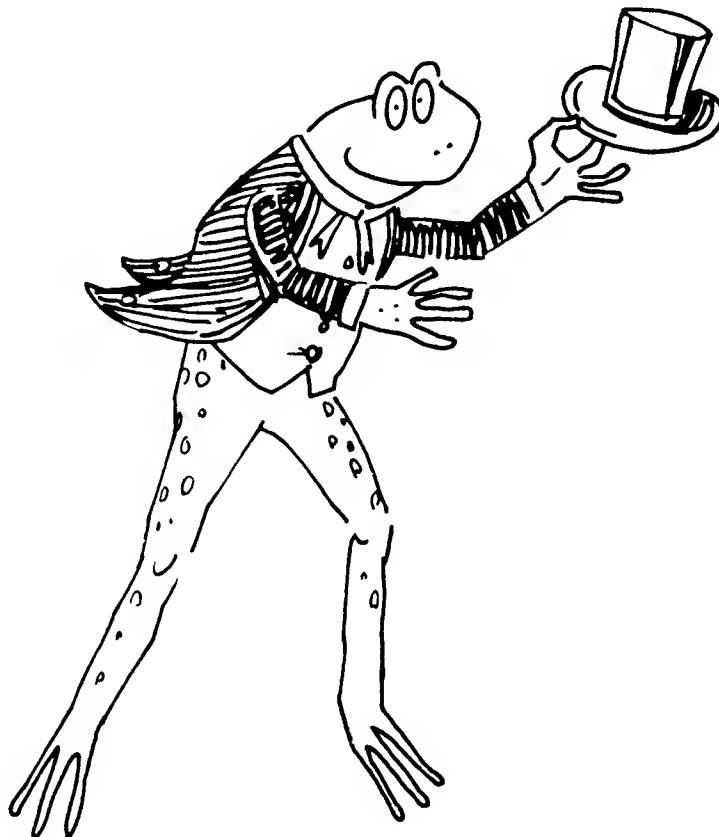
# **LAST DATA HAS BEEN PROCESSED**

- **DISABLE INTERRUPT SYSTEM**
- **CLC ON DEVICE**
- **SET UP 10 MSEC TIMEOUT (STORE -1 IN  
EQT 15)**
- **REENABLE DCPC COMPLETION INTERRUPT**
- **RESTORE MEMORY PROTECT**
- **RESTORE REGISTERS**
- **ENABLE INTERRUPT SYSTEM**
- **RESTORE DMS STATUS**
- **RETURN**



# COMPLETION SECTION

- ENTERED ONLY AFTER TIMEOUT
- SET RETURN PARAMETERS IN A & B REGISTERS
- MODIFY STATUS BITS 0-7 IN EQTS
- RETURN



\*\* RTE DMS PRIVILEGED DRIVER EXAMPLE \*\*

```

0001          ASMB,L,C
0003*
0004 00000      NAM DVYNN      ** RTE DMS PRIVILEGED DRIVER EXAMPLE **
0005          SUP
0006*
0007          ENT IXNN,CXNN
0008*
0009*****
0010* SAMPLE RTE PRIVILEGED DRIVER DVYNN - FOR DMS SYSTEMS *
0011*****
0012*
0013* HANDLES USER PROGRAM REQUESTS TO READ FROM A PRIVILEGED
0014* CONTROLLER
0015*
0016* USER PROGRAM CALLING SEQUENCE:
0017*
0018*     JSB EXEC          CALL EXEC
0019*     DEF **5          RETURN POINT
0020*     DEF RCODE        REQUEST CODE (MUST BE READ REQUEST)
0021*     DEF CUNWD        CONTROL WORD
0022*     DEF BUFFER       ADDRESS OF BUFFER (MUST BE IN SYSTEM COMMON)
0023*     DEF LENTH        LENGTH OF BUFFER
0024*
0025* CAUTION:
0026*
0027* THIS DRIVER WILL NOT WORK WITH MORE THAN ONE PRIVILEGED
0028* CONTROLLER. IF MORE THAN ONE PRIVILEGED CONTROLLER
0029* EXISTS IN A SYSTEM, DVYNN MUST BE
0030* RE-ASSEMBLED WITH ALL NAMES CONTAINING "NN" CHANGED SO
0031* THAT EACH COPY OF THE DRIVER HAS UNIQUE ENTRY POINTS.
0032* THEN ONE DRIVER PER CONTROLLER MUST BE PUT
0033* INTO THE SYSTEM AT GENERATION TIME.
0034*
0035* NOTE:
0036*
0037* 1.) THE DESIGN OF THIS DRIVER ASSUMES THAT THE I/O
0038*     BUFFER BEING PROCESSED IS LOCATED IN SYSTEM COMMON.
0039*     THIS CAUSES THE DRIVER TO BE ENTERED WITH THE
0040*     SYSTEM MAP ENABLED. THIS IS NECESSARY FOR THE
0041*     CORRECT OPERATION OF THE TRAP CELL MODIFICATION
0042*     TECHNIQUE ILLUSTRATED BELOW. IN ADDITION, THE
0043*     BUFFER IN SYSTEM COMMON ALLOWS THE DRIVER TO PUT THE
0044*     DATA VALUES DIRECTLY INTO THE BUFFER, WITHOUT
0045*     THE NEED FOR MAP SWITCHING
0046*
0047* 2.) THIS DRIVER DOES NOT PROCESS POWER FAIL INTERRUPTS.
0048*
0049* 3.) THIS DRIVER DOES NOT PROCESS ANY TIME-OUTS EXCEPT
0050*     FOR THE TIME-OUT THAT IT CREATES AS A MEANS TO
0051*     COMPLETE THE I/O REQUEST AND RETURN TO IOC
0052*

```

\*\* DMS PRIVILEGED DRIVER - INITIATION SECTION \*\*

```

0054*
0055*
0056*          *****
0057*          * INITIATION SECTION *
0058*          *****
0059  00000 000000 IXNN  NOP          INITIATION SECTION ENTRY POINT
0060  00001 072200R      STA SCODE    SAVE SELECT CODE OF CONTROLLER
0061*
0062  00002 066203R      LDB FIRST    ACCESS FIRST TIME THROUGH FLAG
0063  00003 006002      SZB           IS THIS THE FIRST TIME THRU?
0064  00004 026020R      JMP INIT    NO, SO SKIP CONFIGURATION CODE
0065*
0066*  CONFIGURE I/O INSTRUCTIONS
0067*
0068  00005 032217R      IOR LIA      CREATE LIA INSTRUCTION
0069*          :
0070*          :
0071*          :
0072*          :
0073*  MODIFY TRAP CELL
0074*
0075  00006 060000R      LDA $JSB     SET TRAP CELL TO
0076  00007 172200R      STA SCODE,I  JSB $JPNN,I ($JPNN = ADDR OF PXNN)
0077*
0078*  SAVE EQT ADDRESSES
0079*
0080  00010 061774      LDA EQT15     SAVE EQT15
0081  00011 072215R      STA EQ15
0082  00012 061663      LDA EQT4     EQT 4
0083  00013 072214R      STA EQ4
0084  00014 061660      LDA EQT1     AND EQT1
0085  00015 072213R      STA EQ1     ADDRESSES
0086*
0087  00016 002404      CLA,INA       SET FLAG TO PREVENT CONFIGURING ON
0088  00017 072203R      STA FIRST    SUBSEQUENT INITIATIONS
0089*
0090*  CLEAR THE "DRIVER PROCESSES TIME-OUT" BIT TO ALLOW
0091*  NORMAL TIME-OUT OPERATION
0092*
0093  00020 161663  INIT  LDA EQT4,I  ACCESS EQT WORD 4
0094  00021 012221R      AND =B167777  CLEAR BIT 12
0095  00022 171663      STA EQT4,I  AND RESET EQT WORD 4
0096*
0097*  CHECK THE REQUEST CODE
0098*
0099  00023 161665      LDA EQT6,I  ACCESS REQUEST CODE
0100  00024 012222R      AND =B3      ISOLATE REQUEST TYPE
0101  00025 052223R      CPA =B1     READ REQUEST?
0102  00026 026041R      JMP PROC    YES, GO PROCESS READ REQUEST
0103*
0104  00027 052222R      CPA =B3     CONTROL REQUEST?
0105  00030 026033R      JMP CNTRL   YES, GO PROCESS CONTROL REQUEST
0106*
0107  00031 002404      CLA,INA       NO, SO REJECT AS ILLEGAL WRITE REQUEST
0108  00032 126000R      JMP IXNN,I
0109*

```

★★ DMS PRIVILEGED DRIVER - INITIATION SECTION ★★

```

0110* CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
0111* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
0112*
0113 00033 161665 CNTRL LDA EQT6,I      ACCESS CONTROL WORD
0114 00034 012224R      AND =B3700     ISOLATE SUBFUNCTION
0115 00035 002002      SZA              "CLEAR" REQUEST?
0116 00036 026037R      JMP REJCT       NO, SO REJECT AS ILLEGAL CONTROL REQUEST
0117*
0118*      .
0119*      .      EXECUTE CODE TO CLEAR CONTROLLER
0120*      .
0121*
0122 00037 062225R REJCT LDA =B2        REJECT AS ILLEGAL CONTROL REQUEST
0123 00040 126000R      JMP IXNN,I
0124*
0125* SET UP FOR THE DATA TRANSFER
0126*
0127 00041 161667 PROC  LDA EQT8,I      ACCESS # OF CONVERSIONS REQUIRED
0128 00042 003004      CMA,INA          NEGATE FOR CONVERSION COUNTER
0129 00043 072201R      STA CVCTR       AND SAVE
0130 00044 002021      SSA,RSS          REJECT IF
0131 00045 026037R      JMP REJCT       NUMBER <0
0132 00046 161666      LDA EQT7,I      SAVE DATA BUFFER ADDRESS
0133 00047 072202R      STA DAPTR      FOR PXNN
0134*
0135* INITIATE A READ AND RETURN
0136*
0137 00050 016053R      JSB READ        START A READ
0138 00051 103700 1.1  STC SC,C        ENCODE DEVICE
0139 00052 126000R      JMP IXNN,I      RETURN TO IOC
0140*
0141* SUBROUTINE TO INITIATE A READ
0142*
0143 00053 000000 READ  NOP             ROUTINE CONTAINING
0144*      .      CONFIGURED I/O
0145*      .      INSTRUCTIONS TO
0146*      .      SET UP THE DEVICE
0147*      .      TO INITIATE ONE READING
0148 00054 126053R      JMP READ,I

```

★★ DMS PRIVILEGED DRIVER - PRIVILEGED SECTION ★★

```

0150*
0151*
0152*
0153*
0154*
0155* SAVE STATE OF COMPUTER AT INTERRUPT
0156*
0157 00055 000000 PXNN NOP PRIVILEGED SECTION ENTRY POINT
0158*
0159 00056 103100 CLF 0 TURN OFF INTERRUPT SYSTEM
0160*
0161 00057 106706 CLC 6 TURN OFF DCPC COMPLETION INTERRUPTS
0162 00060 106707 CLC 7
0163*
0164 00061 072204R STA ASV SAVE REGISTERS
0165 00062 076205R STB BSV
0166 00063 001520 ERA,ALS
0167 00064 102201 SOC
0168 00065 002004 INA
0169 00066 072206R STA EDSV
0170 00067 105743 STX XSV SAVE X REGISTER
0171 00071 105753 STY YSV SAVY Y REGISTER
0172 00073 105714 SSM DMSTS SAVE DYNAMIC MAPPING SYSTEM STATUS
0173*
0174 00075 061770 LDA MPTFL SAVE OLD MEMORY PROTECT FLAG
0175 00076 072212R STA MPFSV
0176 00077 002404 CLA,INA SET MEMORY PROTECT FLAG TO OFF
0177 00100 071770 STA MPTFL SINCE MEMORY PROTECT IS NOW OFF
0178*
0179 00101 102100 STF 0 TURN INTERRUPT SYSTEM BACK ON
0180*
0181* CHECK FOR SPURIOUS INTERRUPT
0182*
0183 00102 162213R LDA EQ1,I ACCESS REQUEST LIST POINTER WORD
0184 00103 012226R AND #B77777 ISOLATE REQUEST LIST POINTER
0185 00104 002002 SZA IS A REQUEST IN PROGRESS?
0186 00105 026111R JMP PREAD YES, GO PROCESS INTERRUPT
0187*
0188 00106 103100 CLF 0 NO, TURN OFF INTERRUPT SYSTEM
0189 00107 107700 I.2 CLC SC,C RESET CONTROLLER, AND
0190 00110 026121R JMP EXIT IGNORE SPURIOUS INTERRUPT BY RETURNING
0191*
0192* PROCESS READ REQUEST
0193*
0194 00111 PREAD EQU *
0195* . LOAD IN DATA FROM DEVICE
0196* . VIA CONFIGURED I/O INSTRUCTIONS
0197* .
0198*
0199 00111 172202R STA DAPTR,I STORE WORD IN DATA BUFFER
0200 00112 036201R ISZ CVCTR IS THIS THE LAST CONVERSION?
0201 00113 002001 RSS NO
0202 00114 026164R JMP DONE YES, GO SET UP TO TERMINATE CALL
0203*
0204 00115 036202R ISZ DAPTR NO, SET UP FOR NEXT CONVERSION
0205 00116 016053R JSB READ INITIATE IT

```

★★ DMS PRIVILEGED DRIVER - PRIVILEGED SECTION ★★

```

0206*
0207* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
0208*
0209 00117 103100 CLF 0 TURN OFF INTERRUPT SYSTEM TEMPORARILY
0210*
0211 00120 103700 I.3 STC SC,C ENCODE DEVICE
0212*
0213 00121 062212R EXIT LDA MPFSV ACCESS PREVIOUS STATE OF MEMORY PROTECT
0214 00122 002002 SZA WAS MEMORY PROTECT ON?
0215 00123 026134R JMP EXIT1 NO, SO DO NOT TURN ON DCPC INTERRUPTS
0216*
0217 00124 065654 LDB INTBA YES, TURN DCPC COMPLETION INTERRUPTS
0218 00125 160001 LDA B,I BACK ON IF THEY WERE ON INITIALLY.
0219 00126 002020 SSA ON/OFF STATUS IS INDICATED BY BIT 15
0220 00127 102706 STC 6 OF EACH DCPC ASSIGNMENT WORD IN THE
0221 00130 006004 INB INTERRUPT TABLE
0222 00131 160001 LDA B,I
0223 00132 002020 SSA
0224 00133 102707 STC 7
0225*
0226 00134 062206R EXIT1 LDA EOSV RESTORE E AND O REGISTERS
0227 00135 103101 CLO
0228 00136 000036 SLA,ELA
0229 00137 102101 STF 1
0230 00140 066205R LDB BSV RESTORE B-REGISTER
0231 00141 105745 LDX XSV RESTORE X REGISTER
0232 00143 105755 LDY YSV RESTORE Y REGISTER
0233*
0234 00145 062212R LDA MPFSV RESTORE MEMORY PROTECT FLAG
0235 00146 071770 STA MPTFL IN BASE PAGE
0236 00147 002002 SZA WAS MEMORY PROTECT ON AT INTERRUPT?
0237 00150 026157R JMP EXIT2 NO
0238*
0239 00151 062204R LDA ASV YES, RESTORE A-REGISTER
0240 00152 102100 STF 0 TURN ON INTERRUPT SYSTEM
0241 00153 102705 STC 5 SET MEMORY PROTECT ON
0242 00154 105715 JRS DMSTS PXNN,I RESTORE DMS STATUS AND RETURN
0243* (NOTE: EXECUTION OF A "JRS"
0244* INSTRUCTION AFTER TURNING THE
0245* MEMORY PROTECT FENCE ON IS
0246* ALLOWED ONLY IF THE SYSTEM MAP
0247* IS CURRENTLY ENABLED. THIS
0248* DRIVER HAS BEEN DESIGNED SUCH
0249* THAT THIS IS ALWAYS THE CASE.
0250*
0251 00157 062204R EXIT2 LDA ASV NO,RESTORE A-REGISTER
0252 00160 102100 STF 0 TURN ON INTERRUPTS
0253 00161 105715 JRS DMSTS PXNN,I RESTORE DMS STATUS AND RETURN
0254*
0255* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
0256*
0257 00164 103100 DONE CLF 0 TURN OFF THE INTERRUPT SYSTEM
0258 00165 106700 I.4 CLC SC TURN OFF PRIVILEGED DEVICE
0259 00166 003400 CCA SET TIME OUT FOR
0260 00167 172215R STA EQ15,I ONE TICK AND SET
0261 00170 102214R LDA EQ4,I BIT12 IN EQ14 SO

```

★★ DMS PRIVILEGED DRIVER - PRIVILEGED SECTION ★★

0262	00171	032210R	IOR BIT12	RTIOC WILL
0263	00172	172214R	STA EQ4,I	CALL CXNN ON TIME-OUT
0264	00173	026121R	JMP EXIT	GO TO EXIT ROUTINE
0266★				
0267★				
0268★			*****	
0269★			* COMPLETION SECTION *	
0270★			*****	
0271	00174	000000	CXNN NOP	COMPLETION SECTION ENTRY POINT
0272★				
0273	00175	002400	CLA	SET A = 0 = NORMAL RETURN
0274	00176	165667	LDB EQ8,I	SET B = TRANSMISSION LOG
0275	00177	126174R	JMP CXNN,I	RETURN TO IOC
0276★				

\*\* DMS PRIVILEGED DRIVER - DATA AREA \*\*

```

0278*
0279* CONSTANT AND STORAGE AREA
0280*
0281 000000 A EQU 0
0282 000001 B EQU 1
0283 000000 SC EQU 0 DUMMY I/O SELECT CODE NUMBER
0284*
0285 00200 000000 SCODE BSS 1
0286 00201 000000 CVCTR BSS 1
0287 00202 000000 DAPTR BSS 1
0288 00203 000000 FIRST BSS 1
0289 00204 000000 ASV BSS 1
0290 00205 000000 HSV BSS 1
0291 00206 000000 EOSV BSS 1
0292 00207 000000 XSV BSS 1
0293 00210 000000 YSV BSS 1
0294 00211 000000 DMS15 BSS 1
0295 00212 000000 MPFSV BSS 1
0296 00213 000000 EQ1 BSS 1
0297 00214 000000 EQ4 BSS 1
0298 00215 000000 EQ15 BSS 1
0299 00216 010000 BIT12 OCT 10000
0300 00217 102500 LIA LIA 0
0301*
0302* BASE PAGE COMMUNICATIONS AREA DEFINITION
0303*
0304 01650 EQU 1650B
0305 01654 INTHA EQU .+4
0306 01660 ENT1 EQU .+8
0307 01663 ENT4 EQU .+11
0308 01665 ENT6 EQU .+13
0309 01666 ENT7 EQU .+14
0310 01667 ENT8 EQU .+15
0311 01774 ENT15 EQU .+84
0312 01776 MPTEL EQU .+80
0313*
0314* CODE TO SET UP JSB $JPNN,I INSTRUCTION ON BASE PAGE
0315*
0316 00220 000050R $JPNN DEF PXNN PRIV. SECTION ENTRY POINT ADDR
0317*
0318 00000 ORB RESET LOCATION COUNTER TO BASE PAGE
0319 00000 116220R $JSB JSB $JPNN,I JSB INSTR. TO PRIV. SECTION, INDIRECT
0320*
0321 END
** NO ERRORS *TOTAL **RTE ASMB 760924**

```



POWER FAIL



# A POWER

# FAIL

# INTERRUPT

OCCURS WHEN THE PRIMARY POWER DROPS  
BELOW A PREDETERMINED LEVEL.

**IF** a power fail interrupt **THEN**   
it interrupts to location 4 in memory  
the system map is enabled when DMS  
is in the system

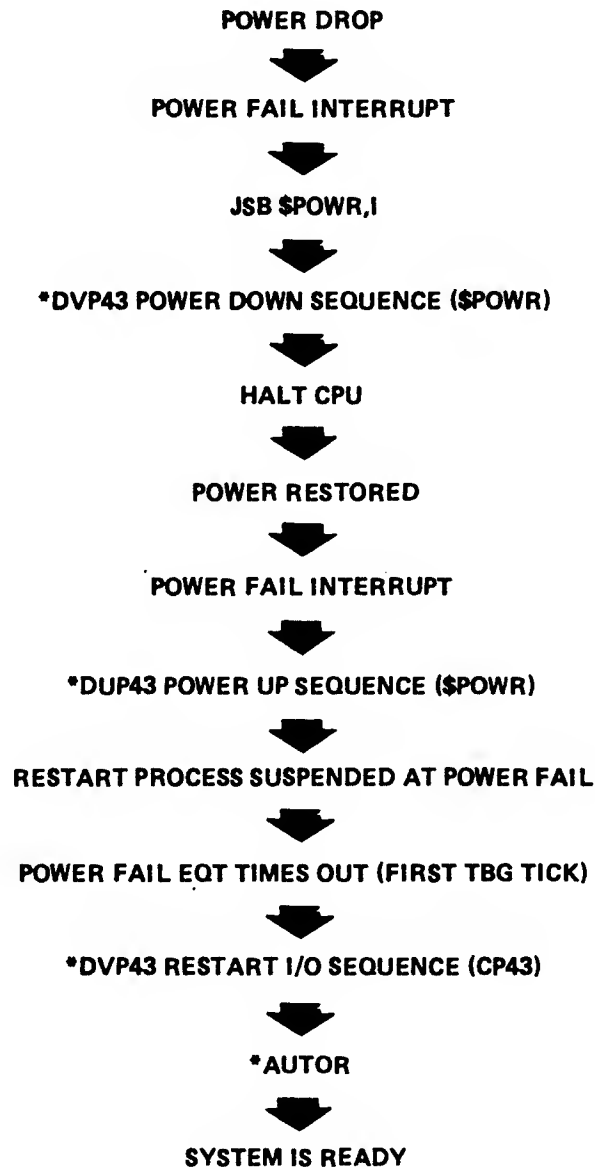
## ON POWER UP

**IF** auto restart switch in main CPU board is in the 'ON' position **THEN** the CPU will resume after ~1/2 sec.

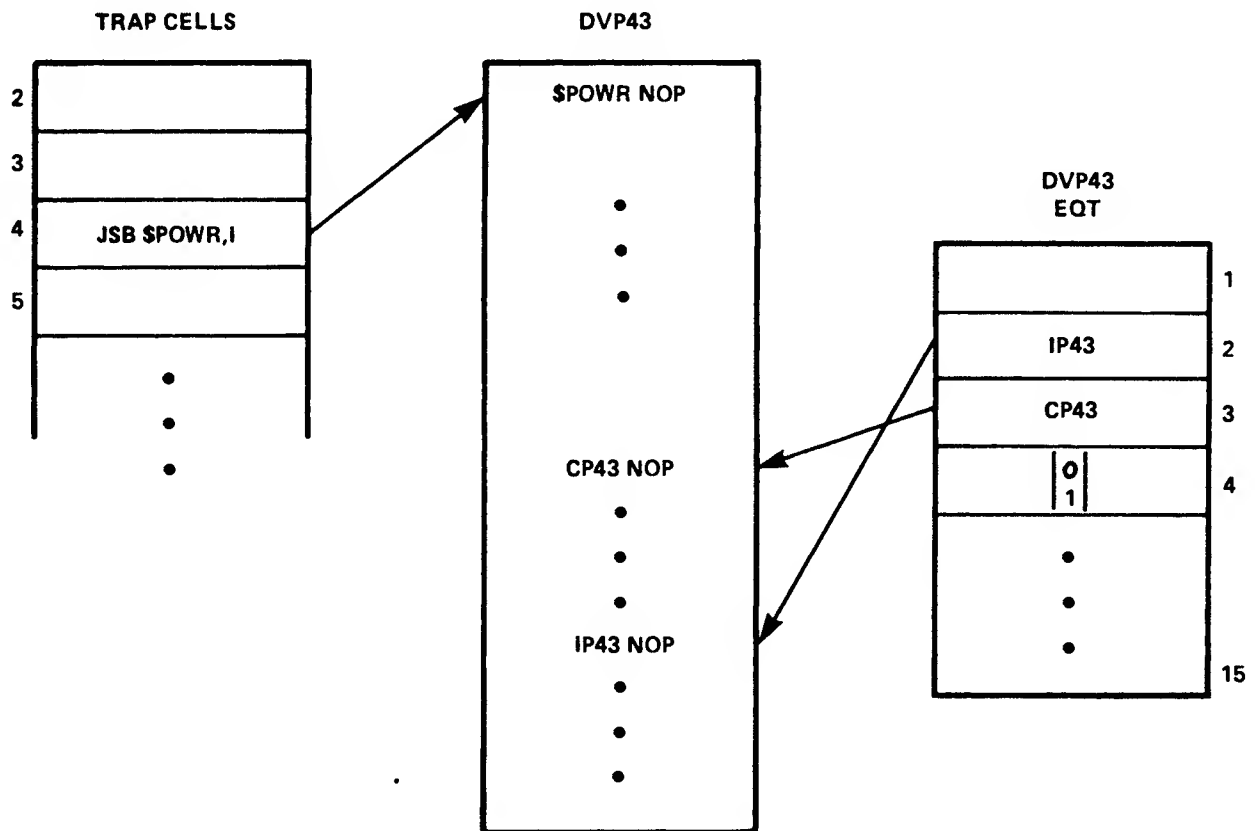
POWER FAIL SYSTEM  
COMPONENTS

- Power-sensing circuits
- Memory sustaining battery (1.75 to 4.25 hours depending on memory size)
- Power fail/automatic restart driver: DVP43 (\$POWER, CP43 and IP43 entries) generated into SDA.
- Automatic restart program (AUTOR)
- Automatic restart switch (ARS)

# POWER FAIL EVENTS



# POWER FAIL LINKAGES



DVP43 POWER DOWN SEQUENCE\*  
(\$POWR)

```

NAM DVP43,0
.
.
.
$POWR NOP          POWER UP/DOWN ENTRY
SFC 4              UP?
JMP UP            YES GO DO UP THING.

JMP DOWN,1        GO TO DOWN ROUTINE
.
.
.
DWN  STF 6B        ABORT DCPC TRANSFERS
STF 7B
STA ASAVE--+
STB BSAVE |
ERA,ALS |
SOC |
INA | ----- SAVE REGISTERS AND INTERRUPT LOCATION
STA EOSAV |
LDA $POWR |
STA PSAVE |
STX XSAVE--+
STY YSAVE
LIA 5-----+
LIB 5 |
CPB 5 | ----- SAVE MP VIOLATION ADDRESS
STA $CIC |
STA $PWR5--+
LIA 2-----+
STA SDMA1 |
LIA 3 | ----- SAVE DCPC WORD COUNTS
STA SDMA2--+
LIA 1          SAVE S REGISTER
STA SSAVE
RSA-----+
STA MEMST |
CLA |
LDB SMAPA | ----- SAVE DMS STATUS AND MAP REGISTERS
LDX MD128 |
XMM -----+
CLC 4          WAIT FOR POWER UP INTERRUPT
HLT 0

```

\* POWER DOWN ROUTINE MUST EXECUTE IN 500 MICRO-SECONDS.

DVP43 POWER UP SEQUENCE  
(\$PCWR)

- Set switch so that another power fail interrupt will halt.
- Reenable power fail hardware
- Restore DMS and MP registers
- Save power fail time
- Set DVP43's time out (EQT entry word 15) to one tick (-1) and set time-out bit (EQT entry word 4).
- Restart system clock (\$SCLK sets the clock up for an immediate interrupt)
- Restore registers and DCPC channels
- Return to point of power fail interrupt



DVP43 RESTART I/O SEQUENCE  
(CPL43)

- For each EQT entry:
  1. If the EQT entry was busy (AV=2) and its power fail bit ("P") set, enter the driver at InXX. The driver's initiator section notes that AV=2 and thus the request is for a power-failure.
  2. If the EQT entry was waiting for a DCPC channel (AV=3), no action is taken.
  3. \$UPIO is called for all other EQT entries to restart requests that were in progress or were pending by calling each driver at InXX.
- Abort and then schedule AUTOR.

## AUTOR

- Call power fail LU to get power fail time
- Sends power fail messages to each LU which is a DVR00 or DVR05 subchannel 0 device.
- Reenable each terminal
- Calls power fail LU a second time to indicate that the recovery process is complete.

## SYSTEM LIBRARY



## LIBRARY SUBROUTINE STRUCTURES

- RE-ENTRANT - TYPE 6
- PRIVILEGED - TYPE 6
- UTILITY - TYPE 7

## MEMORY RESIDENT LIBRARY

- TYPE 6 SUBROUTINES WHICH ARE:
  - A. REFERENCED BY MEMORY RESIDENT PROGRAMS (TYPE 1, 9, 17, OR 25).
  - B. REFERENCED BY OTHER LIBRARY SUBROUTINES (TYPE 6).
- TYPE 14 SUBROUTINES WHICH ARE FORCE LOADED INTO THE MEMORY RESIDENT LIBRARY.

\*\*\*SUBROUTINES IN THE MR LIBRARY ARE ONLY CALLABLE\*\*\*

\*\*\*\*\*FROM PROGRAMS IN THE MR AREA\*\*\*\*\*

## DISC RESIDENT LIBRARY

- TYPE 7 SUBROUTINES
- TYPE 6 SUBROUTINES

## SAMPLE LIBRARY ROUTINES

- PARSE (\$PARS)  
    PARSE AN ASCII STRING USING COMMAS OR DELIMITERS
- CNUMD (\$CVT3) OR CNUMO (\$CVT3)  
    CONVERT A BINARY NUMBER TO ASCII DECIMAL OR ASCII OCTAL
- MESSS  
    ISSUE A SYSTEM COMMAND
- EQLU  
    FIND LU NUMBER OF AN INTERRUPTING DEVICE FROM ITS EQT ADDRESS
- IFBRK  
    TESTS PROGRAMS BREAK FLAG
- INPRS  
    DOES A REVERSE PARSE
- NAMR  
    PARSES A FMGR NAME
- IFTTY  
    DETERMINES IF LU IS INTERACTIVE OR NOT
- LOGLU  
    RETURNS LU OF TERMINAL THAT SCHEDULED PROGRAM



## UTILITIES



## UTILITY PROGRAMS

- LOCUS - library of contributed user software
- Software Service Kit

MAPIO(LUPRN)

-----

CONTRIBUTION #: 598  
CLASS: 103  
CONTRIBUTOR: LARRY SMITH  
HP - NEELY SANTA CLARA

PART NUMBER: 22682-18919  
PRICE: \$20  
DATE CODE: 1617  
LANG: ASSEMBLY RELOCATABLE  
OP SYS: RTE

MAPIO

THIS PROGRAM PRINTS A COMPACT TABLE BY LOGICAL UNIT OF ANY RTE I/II OR III I/O CONFIGURATION. THE MAP INCLUDES THE LU, EQT, SELECT CODE, SUB-CHANNEL AND CHANNEL, EQT ADDRESS, DRIVER ADDRESS, AND DEVICE TYPE. THE OUTPUT CAN BE SENT TO ANY DEVICE AND DEVICE NAMES CAN BE CHANGED BY THE USER.

HARDWARE REQUIREMENTS: TERMINAL OR PRINT DEVICE

ORDER #22682-13319

SOURCE ON CASSETTE

\$35.00

\*\*\*\*\* DOCUMENTATION FILE \*\*\*\*\*

\*\*\*\*\* SOFTWARE SERVICE KIT \*\*\*\*\*

# MATERIAL LIST

PART #	REV	DESCRIPTION	
24999-16048	1727	JSAVE DK LU UTIL	
24999-16049	1727	JRSTR DK LU UTIL	
24999-16050	1651	SDLS4 DISK UTIL	BINARIES
24999-16051	1712	MXREF XREF ON BN	
24999-16052	1651	CMM3 MEM/DK MOD	
24999-16053	1646	FGETR GETR FILE	
24999-16055	1651	CLASS I/O UTIL	
24999-16044	1650	RECON	
24999-16163	1727	JVRFY DK LU UTIL	
24999-16167	1731	MLOAD RELOCATABLE	
24999-16168	1731	MDUMP RELOCATABLE	
24999-16171	1736	LTAT RELO - LIST TRK ASSGN TABLE.	
24999-18065	1727	JSAVE DK LU UTIL	
24999-18066	1727	JRSTR DK LU UTIL	
24999-18067	1651	SDLS4 DISK UTIL	
24999-18068	1712	MXREF XREF ON BN	
24999-18069	1651	CMM3 MEM/DISK MOD	
24999-18070	1646	FGETR GET FILE	SOURCES
24999-18083	1651	CLASS I/O UTIL	
24999-18052	1650	RECON	
24999-18071	1727	KIT DOCUM.	
24999-18163	1727	JVRFY DK LU UTIL	
24999-18167	1731	&MLOAD SOURCE	
24999-18168	1731	&MDUMP SOURCE	
24999-16171	1736	&LTAT SOURCE LIST TRK ASSGN TABLE.	

# TABLE OF CONTENTS

CMM3	-	MEMORY ACCESS AND MODIFICATION
SDLS4	-	READ FROM CUPERTINO DISTRIBUTION DIRECTLY INTO RTE FMP FILES
JSAVE	-	SAVE DISC CARTRIDGE ON MAG TAPE
JRSTR	-	RESTORE DISC CARTRIDGE FROM JSAVE MAG TAPE
JVRFY	-	VERIFY JRSTR DISC WITH TAPE & JSAVE TAPE WITH DISC
FGETR	-	ACCESS FILES AND DIRECTORY LIST ON JSAVE MAG TAPE
MXREF	-	CROSS REFERENCE MAP LISTING
CLASS	-	DISPLAY STATUS OF CLASS TABLE, LIST CONTENTS, OR CLEAR PENDING BUFFERS
RECON	-	BOOTSTRAP RECONFIGURATION FOR GRANDFATHER DISCS
&MLOAD	-	LOADS SYSTEM MEMORY MAG TAPE TO 21MX CPU
&MDUMP	-	DUMPS SYSTEM MEMORY (0 TO 77777B) TO MAG TAPE
&LTAT	-	LISTING THE TRACK ASSIGNMENT TABLE

## PERFORMANCE MEASUREMENT





## TYPES OF PROCESSES TO MEASURE

### I. PROCESSES WITHOUT WAIT

examples: library functions (SIN, COS, etc.)  
obtaining system time  
locking an LU  
going privileged  
scheduling a son program

### II. PROCESSES WITH WAIT

examples: I/O transfers  
scheduling of disc resident programs

## RTE OVERHEAD

- CONSISTS OF RTE TIME TO SERVICE TBG INTERRUPTS
- MEASURED BY EXECUTING A FIXED NUMBER OF INSTRUCTIONS IN A KNOWN TIME (F.TIME) AND CALCULATING THE ELAPSED TIME (E.TIME) OF THE INSTRUCTIONS
- RTE OVERHEAD CAN THEN BE CALCULATED:

$$\text{TBG\%} = \frac{\text{E.TIME} - \text{F.TIME}}{\text{E.TIME} \times 100}$$

- SAMPLE RTE-II OVERHEAD TIMES:

2100A - 1.43%  
M SERIES - 3.31%  
E SERIES - 1.21%

# PROGRAM TO MEASURE RTE OVERHEAD TIME

```

0001 FTN4,L
0002 C      6/03/76 WEIMAN
0003 C      PROGRAM SYSOH
0004 C
0005 C
0006 C      MEASURES BASIC SYSTEM OVERHEAD ON RTE SYSTEMS.
0007 C      MAY BE RUN ALONE, OR USED IN CONJUNCTION WITH
0008 C      ANOTHER PROGRAM TO MEASURE THE SYSTEM OVERHEAD
0009 C      THAT PROGRAM INTRODUCES.
0010 C
0011 C
0012 C

0013 C      SCHEDULE PARAMETERS:
0014 C          #1-TTY LU FOR MESSAGE OUTPUT.
0015 C          #2-CPU: <2 = 2100
0016 C              2 = 21MX
0017 C              >2 = 21XE
0018 C
0019 C      SEQUENCE OF OPERATIONS:
0020 C          1)GET START TIME
0021 C          2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
0022 C          3)GET FINISHED TIME.
0023 C          4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME AND
0024 C             EXECUTION TIME, AS A PERCENTAGE OF ELAPSED TIME.
0025 C
0026 C      INTEGER STIME(S),FTIME(S),IPRAM(S),IOFF,NTIME,LU
0027 C      INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
0028 C      INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
0029 C      INTEGER CPU
0030 C      REAL XTIME
0031 C      EXECUTION TIME = DATA STORED IN "XTIME"
0032 C      EQUIVALENCE (IPRAM(1),LU)
0033 C      EQUIVALENCE (IPRAM(2),CPU)
0034 C      EQUIVALENCE (IPRAM(3),NCHAR)
0035 C      EQUIVALENCE (FTIME(1),FTIME1)
0036 C      EQUIVALENCE (FTIME(2),FTIME2), (FTIME(3),FTIME3)
0037 C      EQUIVALENCE (FTIME(4),FTIME4), (FTIME(5),FTIME5)
0038 C      EQUIVALENCE (STIME(1),STIME1), (STIME(2),STIME2)
0039 C      EQUIVALENCE (STIME(3),STIME3), (STIME(4),STIME4)
0040 C      EQUIVALENCE (STIME(5),STIME5)
0041 C      DATA XTIME/56.85568/
0042 C
0043 C
0044 C
0045 C      1      FORMAT(" CPU IS 2100A")
0046 C      2      FORMAT(" CPU IS 21MX")
0047 C      3      FORMAT(" CPU IS 21XE")
0048 C
0049 C      GET SCHEDULE PARAMETERS
0050 C      CALL PMPAR(IPRAM)
0051 C      DEFAULT TTY-LU
0052 C      IF(LU .LT. 1) LU=1
0053 C
0054 C      USE PROPER EXECUTION TIME FOR COMPUTER BEING USED.
0055 C
0056 C      IF(CPU .LT. 2) WRITE(LU,1)
0057 C      IF(CPU .LT. 2) XTIME=56.85568
0058 C      IF(CPU .EQ. 2) XTIME=64.40780
0059 C      IF(CPU .EQ. 3) WRITE(LU,2)
0060 C      IF(CPU .GT. 2) WRITE(LU,3)
0061 C      IF(CPU .GT. 2) XTIME=34.91070
0062 C
0063 C
0064 C
0065 C      GET START TIME
0066 C      500    CALL EXEC(11,STIME)
0067 C      LOOP
0068 C      DO 1000 I=1,1000
0069 C      DO 1000 J=1,1000
0070 C      DO 1000 L=1,3
0071 C      1000    CONTINUE
0072 C      GET FINISHED TIME.
0073 C      CALL EXEC(11,FTIME)
0074 C
0075 C      COMPUTE ELAPSED TIME
0076 C
0077 C      ETIME=(FTIME/STIME)*.01 +(FTIME2-STIME2) +
0078 C      1 (FTIME3-STIME3)*63. +(FTIME4-STIME4)*3600.
0079 C      IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
0080 C
0081 C      PRINT ELAPSED TIME CPU LOAD AS PERCENTAGE
0082 C      OF ELAPSED TIME.
0083 C      2000    FORMAT(" ELAPSED TIME="F8.2"SECS.CPU LOAD="F8.3"%")
0084 C      CPULOD=(ETIME-XTIME)/ETIME *100.0
0085 C      WRITE(LU,2000) ETIME, CPULOD
0086 C      2400    CONTINUE
0087 C
0088 C
0089 C      2700    CONTINUE
0090 C      CALL EXEC(3,1100B+LU,-1)
0091 C      3000    END

```

## MEASURING PROCESSES WITHOUT WAIT

- WRITE A PROGRAM TO EXECUTE THE PROCESS (SIN,COS, ETC.) A LARGE NUMBER OF TIMES (1,000 TO 10,000)
- RECORD THE ELAPSED TIME OF EXECUTING THE PROCESS
- PROCESS SERVICE TIME WILL EQUAL:

$$\frac{\text{ELAPSED TIME (1-TBG\%/100)}}{\text{NUMBER OF EXECUTION TIMES}}$$

## MEASURING PROCESSES WITH WAIT

- WRITE TWO PROGRAMS:

OVRHD program executes in a fixed amount of time  
PROC program to repeatedly perform the process

- ENSURE THAT EACH PROGRAM WILL HAVE A SEPARATE PARTITION AND THAT PRIORITY OF "PROC" > "OVRHD"
- THE PROGRAMS ARE RUN SIMULTANEOUSLY WITH "OVRHD" RUNNING WHENEVER "PROC" IS WAITING
- WHEN "OVRHD" COMPLETES, IT RECORDS ELAPSED TIME AND "PROC" RECORDS NUMBERS OF PROCESSES COMPLETED
- PROCESS SERVICE TIME WILL EQUAL:

$$\frac{\text{ELAPSED TIME}(1-\text{TBG\%/100}) - \text{FIXED TIME}}{\text{NUMBER OF EXECUTION TIMES}}$$

EXAMPLE MEASUREMENT OF A PROCESS  
WITH WAIT

Suppose we need to know the CPU time consumed outputting characters to a terminal:

1. Program OVRHD is shown on page 24-7.
2. Process program T0002 is shown on page 24-8.
3. System common is used to:
  - Count number of EXEC 2 calls (T0002)
  - Set a start flag (T0002)
  - Set a stop flag (OVRHD)
4. Program OVRHD schedules program T0002.
5. When OVRHD completes, the time for each EXEC 2 call will equal:

$$\frac{\text{ELAPSED TIME}(1-\text{TBG\%}/100) - \text{FIXED TIME}}{\text{NUMBER OF EXEC 2 CALLS}}$$

# OVRHD PROGRAM

```

0001 FTM4,L
0002 C S/17/76 WEIMAN
0003 PROGRAM OVRHD
0004 COMMON IBUSY,ICOUNT,ICNTR2,IOPTR,ISTOP
0005 C
0006 C
0007 C PROGRAM TO MAKE SYSTEM OVERHEAD MEASUREMENTS
0008 C ON RTE-11 SYSTEMS
0009 C COMMON COMMUNICATION: IBUSY=FLAG, SET BY
0010 C OVRND WHEN IT WANTS THE FOREGROUND PROGRAM
0011 C TO DO SOMETHING. IT IS CLEARED WHEN THAT
0012 C TASK IS DONE.
0013 C ICOUNT,ICNTR2 FORM A TWO-WORD COUNTER
0014 C IOPTR=NUMBER OF WORDS/CHARACTERS
0015 C
0016 C
0017 C SCHEDULE PARAMETERS:
0018 C #1-TTY LU FOR MESSAGE OUTPUT.
0019 C #2-CPU: <2 = 2100
0020 C 2 = 21MX
0021 C >2 = 21XE
0022 C #3= NUMBER OF CHARACTERS PRINTED.
0023 C * = WORDS, --CNARS.
0024 C SEQUENCE OF OPERATIONS:
0025 C 1)GET START TIME
0026 C 2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
0027 C 3)GET FINISHED TIME.
0028 C 4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME
0029 C AND EXECUTION TIME, AS A PERCENTAGE OF
0030 C ELAPSED TIME.
0031 C INTEGER STIME(S),FTIME(S),IPRAM(S),LU
0032 C INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
0033 C INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
0034 C INTEGER IPRDG(3)
0035 C INTEGER CPU
0036 C REAL CLOCK,XTIME
0037 C EXECUTION TIME = DATA STORED IN "XTIME"
0038 C EQUIVALENCE (IPRAM(1),LU)
0039 C EQUIVALENCE (IPRAM(2),CPU)
0040 C EQUIVALENCE (IPRAM(3),NCHAR)
0041 C EQUIVALENCE (FTIME(1),FTIME1)
0042 C EQUIVALENCE (FTIME(2),FTIME2), (FTIME(3),FTIME3)
0043 C EQUIVALENCE (FTIME(4),FTIME4), (FTIME(5),FTIME5)
0044 C EQUIVALENCE (STIME(1),STIME1), (STIME(2),STIME2)
0045 C EQUIVALENCE (STIME(3),STIME3), (STIME(4),STIME4)
0046 C EQUIVALENCE (STIME(5),STIME5)
0047 C DATA XTIME/56.85568/
0048 C
0049 C SET THE FOREGROUND PROGRAM'S NAME
0050 C
0051 C DATA IPRDG/2HT0,2N00,2N2 /
0052 C
0053 C
0054 C GET SCHEDULE PARAMETERS
0055 C CALL PMPAR(IPRAM)
0056 C 1 FORMAT(" CPU IS 2100A")
0057 C 2 FORMAT(" CPU IS 21MX")
0058 C 3 FORMAT(" CPU IS 21XE")
0059 C DEFAULT TTY LU
0060 C IF(LU .LT. 1) LU=1
0061 C IOPTR=NCHAR
0062 C 401 FORMAT(" # CHARACTERS="IS)
0063 C WRITE(LU,401) IOPTR
0064 C
0065 C USE PROPER EXECUTION TIME FOR COMPUTER BEING
0066 C USED.
0067 C IF(CPU .GE. 2) GOTO 5
0068 C COMPUTER IS 2100
0069 C WRITE(LU,1)
0070 C XTIME=56.85568
0071 C GOTO 15
0072 C CONTINUE
0073 C IF(CPU .GT. 2) GOTO 6
0074 C COMPUTER IS 21MX
0075 C WRITE(LU,2)
0076 C XTIME=64.40780
0077 C GOTO 15
0078 C CONTINUE

0079 C COMPUTER IS 21MX-E SERIES
0080 C WRITE(LU,3)
0081 C XTIME=34.91070
0082 C 15 CONTINUE
0083 C
0084 C
0085 C SCHEDULE "SLAVE" TASK PROGRAM
0086 C
0087 C IBUSY=0
0088 C ISTOP=0
0089 C 100 CALL EXEC(10,IPROG,IOPTR)
0090 C
0091 C WAIT FOR IT TO COME IN FROM THE DISC
0092 C
0093 C IF(IBUSY .EQ. 0) GOTO 100
0094 C
0095 C IF PROGRAM HAS ALREADY FINISHED, SKIP WAIT
0096 C LOOP.
0097 C IF(IBUSY .EQ. -2) GOTO 1050
0098 C
0099 C
0100 C GET START TIME
0101 C 500 CALL EXEC(11,STIME)
0102 C LOOP
0103 C DO 1000 I=1,1000
0104 C DO 1000 J=1,1000
0105 C DO 1000 L=1,3
0106 C 1000 CONTINUE
0107 C GET FINISHED TIME.
0108 C CALL EXEC(11,FTIME)
0109 C GET # OF COUNTS
0110 C
0111 C CONVERT NUMBER USING "ICOUNT" AS LOW 16 BITS,
0112 C AND "ICNTR2" AS HIGH 15 BITS.
0113 C
0114 C 1050 CONTINUE
0115 C I1J=ICOUNT
0116 C I1K=ICNTR2
0117 C
0118 C IF PROGRAM ALREADY STOPPED, SKIP WAIT
0119 C
0120 C IF(IBUSY .EQ. -2) GOTO 1020
0121 C
0122 C SIGNAL PROGRAM TO TERMINATE
0123 C
0124 C ISTOP=-1
0125 C 1010 IF(ISTOP .GE. 0) GOTO 1010
0126 C
0127 C CONVERT COUNTERS
0128 C
0129 C 1020 CONTINUE
0130 C I1J=IAND(I1J,77777B)
0131 C FTM=I1J
0132 C IF(I1J .LT. 0) FIN=FIN+32768.0
0133 C FIN=FIN+I1K*65536.0
0134 C
0135 C COMPUTE ELAPSED TIME
0136 C
0137 C FTIME=(FTIME-STIME)*.01 +(FTIME2-STIME2) *
0138 C 1 (FTIME3-STIME3)*60. +(FTIME4-STIME4)*3600.
0139 C IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
0140 C
0141 C PRINT ELAPSED TIME,CPU LOAD AS PERCENTAGE
0142 C OF ELAPSED TIME, AND NUMBER OF EVENTS.
0143 C 2000 FORMAT(" ELAPSED TIME="F8.2"SFCS.CPU
0144 C LOAD="F6.3 "% "EVENTS="F8.0)
0145 C CPULOD=(ETIME-XTIME)/ETIME *100.0
0146 C WRITE(LU,2000) ETIME,CPULOD,FIN
0147 C 2450 CONTINUE
0148 C
0149 C
0150 C
0151 C CALL EXEC(3,1100B+LU,-1)
0152 C END

```

# T0002 PROGRAM

```

0001          ASMB,L
0002 00000      NAM T0002,2,70 S 17 76 3:30 PM
0003          SUP
0004          COM BUSY,COUNT,CNTR2,IQPT,ISTOP
0005          EXT EXEC
0006 00000      T0002 EQU *
0007 00000 002400 CLA          ZERO THE
0008 00001 072001C STA COUNT  COUNTER
0009 00002 072002C STA CNTR2
0010 00003 002004 INA
0011 00004 072000C STA BUSY
0012 00005      LOOP EQU *
0013 00005 062004C LDA ISTOP
0014 00006 002020 SSA
0015 00007 026023R JMP STOP
0016 00010 016001X JSB EXEC  OUTPUT SOME
0017 00011 000016R DEF **5
0018 00012 000032R DEF D2      CHARACTERS
0019 00013 000031R DEF D1
0020 00014 000033P DEF MSG
0021 00015 000003C DEF IQPT
0022 00016 036001C ISZ COUNT  INCREMENT THE COUNTER
0023 00017 026005R JMP LOOP
0024 00020 036002C ISZ CNTR2  ROLLED OVER. BUMP
                                OTHER COUNTER
0025 00021 000000 NOP
0026 00022 026005R JMP LOOP
0027*
0028 00023      STOP EQU *
0029 00023 002400 CLA          SET "STOPPED"
0030 00024 072004C STA ISTOP  FLAG
0031 00025 016001X JSB EXEC  TERMINATE
0032 00026 000030R DEF **2
0033 00027 000030R DEF D6
0034 00030 000006 D6 DEC 6
0035 00031 000001 D1 DEC 1
0036 00032 000002 D2 DEC 2
0037 00033 040523 MSG ASC 28,ASASDFASDFASDFASDFASDF
                                ASDFASDFASDFASDF
0038          END T0002
** NO ERRORS *TOTAL **RTE ASMB 760924**

```



SAMPLE RTE-IV PERFORMANCE  
MEASUREMENT

MEASUREMENT  
-----

E-SERIES, 7905, HS-MEMORY  
-----

Program Schedule

INTERR->MR	1.655 ms.
MR->MR	2.216
BG->MR	2.416
BG->BG	3.114
BG->EMA	3.304

TBG Overhead	295 usec.
(0 programs in T.L.&20EQTs)	

Overhead to go Privileged	290 usec.
---------------------------	-----------

Max. I/O Throughput (500 word buffer)	UB(B)
Standard	3,554(3,581) words/sec
DCPC	158,227(112,612)
Privileged	6,251(12,518)

-----  
\* See SE Note 101(4-18-78). For RTE-II see SA Note 156

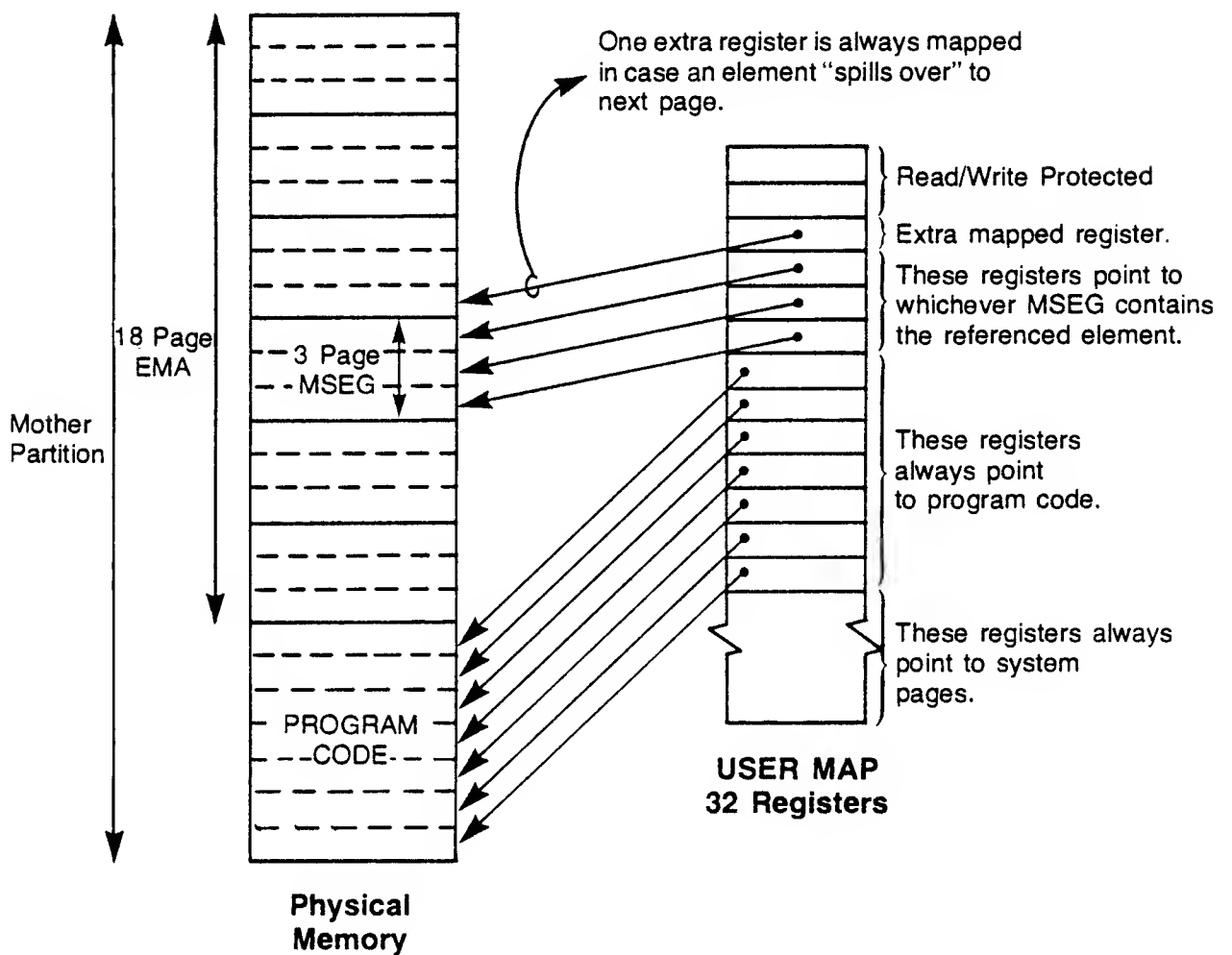


# **EXTENDED MEMORY AREA**



# EMA IN A NUTSHELL

- User map registers change to point to different physical pages as needed to reference data.



$$(\text{System Pages} + \text{Program Size} + \text{MSEG size} + \text{EXTRA PAGE}) \leq 32.$$

## PROGRAMMING WITH EMA

### USING EMA IN FORTRAN PROGRAMS

\$EMA( blockname , mseg )

The \$EMA statement listed above must be the first non-comment statement of the program. The "\$" must appear in column one.

blockname is the name of a common block to be further defined in a named COMMON statement.

mseg is the size in pages of the MSEG. Specify 0 to default this size to the largest possible.

### EXAMPLE EMA DECLARATION

FTN4, L

\$EMA(XYZ, 0)

PROGRAM EXMPL

COMMON/XYZ/IA(1000,100),IB(32767),REAL(10000)

The above declarations allocate 132,767 words for integer variables and 20000 words for real storage for a total of 152,767 words in EMA. The MSEG size will be defaulted.

### EMA SIZE

- Refers to number of pages of physical memory necessary to contain all EMA data.
- EMA size for FORTRAN programs is set by the compiler.  
EMA size = total number of pages necessary to contain all variables declared to be in EMA.

### EXAMPLE

```
EMA SIZE = 32 PAGES;MSEG size = default
FTN4,L
$EMA(ABC,0)
      PROGRAM EMA2
      COMMON/ABC/IA(20000),IB(12000)
```

```
      .
      .
      .
      .
```

## MSEG

- MSEG refers to the pages of EMA that are currently mapped into a program's logical address space. The MSEG is the "window" of data currently accessible to the program under the current map registers.
- MSEG size is defined in a FORTRAN program with the following statement, which should be the first non-comment statement:

    \$EMA (blockname, mseg)

where

blockname is a named common block further defined later  
                    in the program

mseg is the MSEG size

### EXAMPLE

FTN4,L

\$EMA(XYZ,2)

    PROGRAM EMA1

    COMMON/XYZ/IA(1000,200), IB(20000)

    .  
    .  
    .

This defines arrays IA and IB to be in EMA.MSEG size equals 2.



# DEFAULTING MSEG

- MSEG size can be defaulted by specifying 0.

Largest MSEG possible is used then.

All user map registers not pointing to program and system are used to point to MSEG.

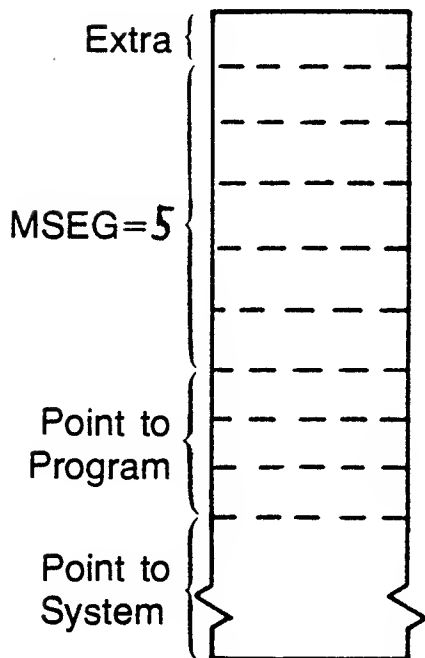
If default used, MSEG size can be modified on-line with SZ command.

Default value is set at load time.

Default value =  $(32 - \text{\#mapped system pages} - \text{program size} - 1)$ .

`$EMA(XYZ,0)`

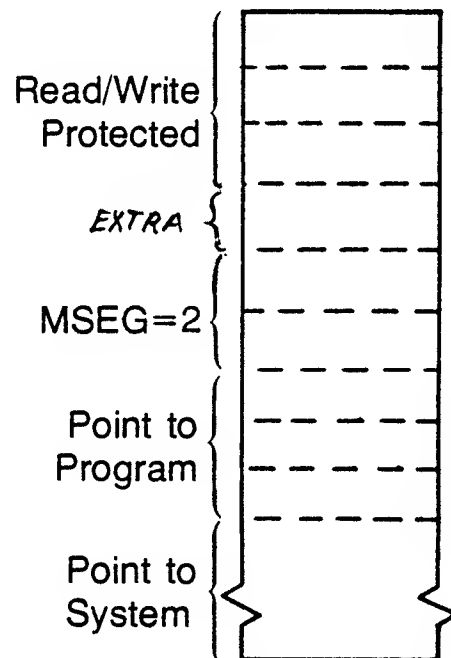
MSEG Default (set at  
load time)



**USER MAP  
REGISTERS**

`$EMA(XYZ,2)`

MSEG Specified as 2



**USER MAP  
REGISTERS**

## CONSIDERATIONS OF MSEG SIZE

- In an all-FORTRAN program executing with the EMA firmware, MSEG size makes absolutely no difference.
- The EMA firmware always maps two pages (one page containing the data and one extra page). Therefore the MSEG size makes no difference.
- The Assembly Language programmer may want to specify MSEG size since subroutines .EMIO and MMAP use the MSEG size.

## LOADING/UNLOADING DATA INTO EMA IN FORTRAN

- To load/unload data into an EMA array, load a non-EMA buffer in the user program, then copy the buffer into/out of EMA. See the example program on the following pages.
- Use formatted or unformatted READ/WRITE statements
- Neither FMP or EXEC calls may be made in FORTRAN with EMA variables.

## PASSING EMA VARIABLES TO SUBROUTINES

- EMA variables must be passed to subroutines using call-by-value.

Call-by-value implies the subroutine cannot modify the actual parameter since the subroutine is passed the value and not the address of the variable.

Call-by-value is indicated in one of two ways:

- 1) Enclosing the actual parameter in an extra layer of parentheses

CALL FUN((UA(2)))

- 2) Making the actual parameter part of an arithmetic expression

CALL FUN(IA(2)+0)

- Subroutines can only modify EMA variables by declaring the same EMA variables and then accepting the subscripts as parameters. See the example on the following pages.

&TEST T=000004 IS ON CR00002 USING 00008 BLKS R=0044

```
0001 C*****
0002 C THIS PROGRAM LOADS CONSECUTIVE INTEGERS INTO A 1-D EMA
0003 C ARRAY. THEN IT LOADS DATA INTO A 2-D EMA ARRAY OFF THE
0004 C DISC. THEN IT CALLS A SUBROUTINE TO SQUARE THE (100,125)
0005 C ELEMENT OF THE 2-D ARRAY.
0006 C*****
0007 FTN4,L
0008 SEMA(XYZ,0)
0009 PROGRAM TEST
0010 COMMON/XYZ/IA(20000),IB(128,150)
0011 DIMENSION IBUF(128),IDCB(144),NAME(3)
0012 DATA NAME/2HFI,2HLE,2HXX/
0013 C *****
0014 C LOAD CONSECUTIVE INTEGERS INTO THE 1-D ARRAY
0015 DO 10 J=1,20000
0016 IA(J)=J
0017 10 CONTINUE
0018 C *****
0019 C OPEN IDCB AND READ DATA FROM DISC INTO SUCCESSIVE
0020 C COLUMNS OF 2-D ARRAY.
0021 CALL OPEN(IDCB,IERR,NAME)
0022 IF (IERR .LT. 0) GO TO 9000
0023 DO 30 K=1,150
0024 CALL READF(IDCB,IERR,IBUF)
0025 IF (IERR .LT. 0) GO TO 9100
0026 DO 20 L=1,128
0027 IB(L,K)=IBUF(L)
0028 20 CONTINUE
0029 30 CONTINUE
0030 C *****
0031 C CALL SUBROUTINE TO SQUARE THE (100,125) ELEMENT OF 2-D ARRAY
0032 CALL SQRE(100,125)
0033 GO TO 9999
0034 C *****
0035 9000 WRITE(LU,9010) IERR
0036 9010 FORMAT(" /TEST: OPEN ERROR,IERR=",I6)
0037 GO TO 9999
0038 C *****
0039 9100 WRITE(LU,9110) IERR
0040 9110 FORMAT(" /TEST: READF ERROR,IERR=",I6)
0041 GO TO 9999
0042 C *****
0043 9999 CALL CLOSE(IDCB)
0044 END
```

&SQRE T=00004 IS ON CR00002 USING 00002 BLKS R=0005

```
0001  FTN4,L
0002  $EMA(XYZ,0)
0003      SUBROUTINE SQRE(ISUB1,ISUB2)
0004      COMMON/XYZ/IA(20000),IB(128,150)
0005      IB(ISUB1,ISUB2)=IB(ISUB1,ISUB2)*IB(ISUB1,ISUB2)
0006      END
```

# **EMA IN ASSEMBLY LANGUAGE**

## EMA IN ASSEMBLY LANGUAGE

- Why use Assembly Language EMA?
- EMA Statement
- EMA Subroutines
- EMAST - Returns Information about EMA
- MMAP - Maps Physical Pages Into Logical Address Space
- Table Definition for .EMIO and .EMAP
- .EMAP - Resolves References to EMA Elements
- .EMIO - Used for I/O from EMA Arrays



## WHY USE ASSEMBLY LANGUAGE EMA?

- Increase speed by doing mapping only when necessary.
- Use EXEC reads and writes to perform I/O quickly to/from EMA arrays.
- Call .EMIO to do fast I/O for non-standard buffers.

## EMA STATEMENT

The following statement is required to use EMA in Assembly Language:

```
label EMA m1,m2
```

where:

label = the name assigned to the EMA area (may only be referenced directly; indirects and offsets are not allowed).

m1 = EMA size in pages  $.0 \leq m1 \leq 1023$ . Specifying 0 defaults EMA size.

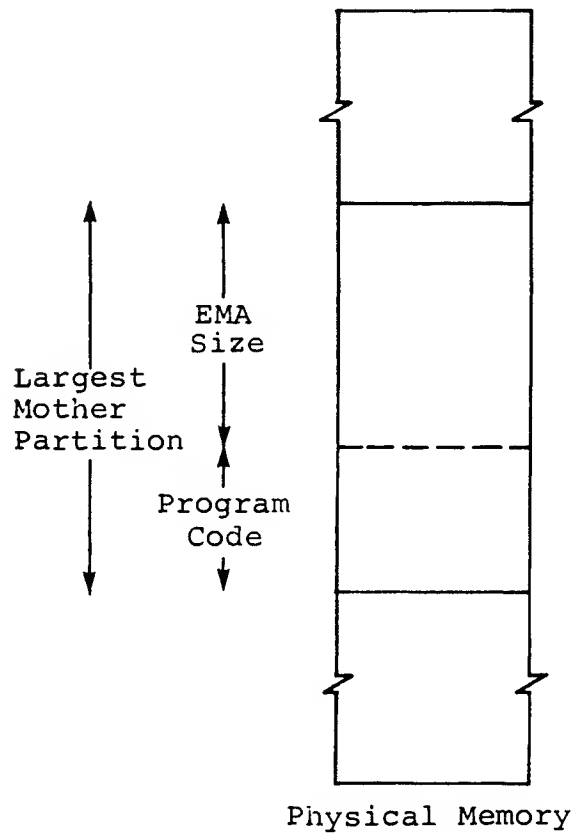
m2 = MSEG size.  $.0 \leq m2 \leq 31$ . Specifying 0 defaults MSEG size.

- Only one EMA pseudo-op per program is allowed.
- References to EMA labels are processed as indirect addresses through a base page link at load time. This is similar to external references, except EMA labels can't be used with indirect or offset.

Refer to the RTE-IV Assembler Manual for more information.

### DEFAULTING EMA SIZE

- If the EMA size is defaulted, it will be set at dispatch time to the size of the largest mother partition in the system. This default may be taken only for Assembly Language programs.

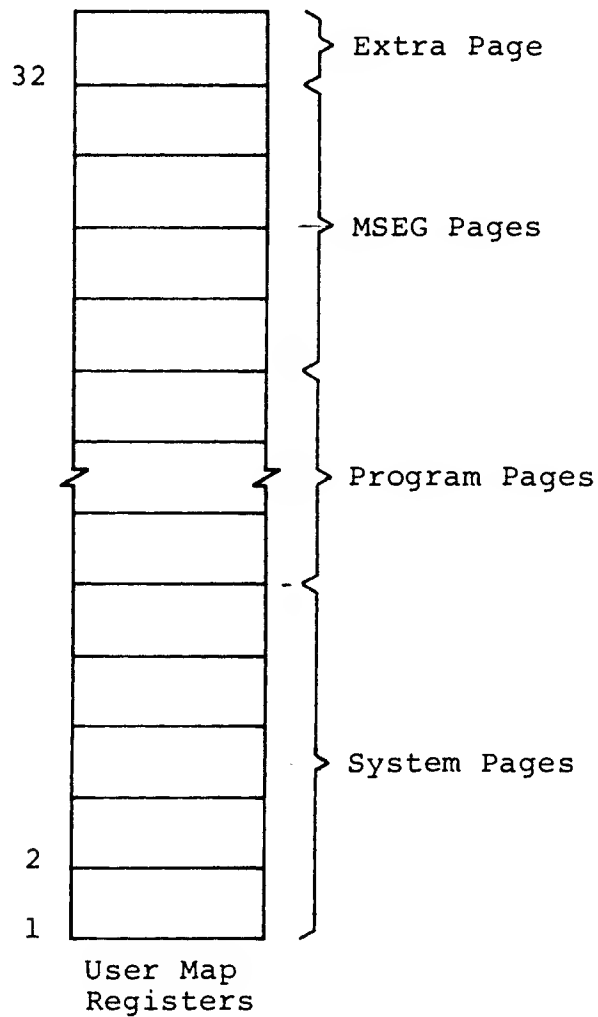


### DEFAULTING MSEG SIZE

- If the MSEG size is defaulted, it is set at load time according to the following formula:

Default MSEG = 32-System Pages-Program Pages-1

- This allocates all the remaining registers to the MSEG except one that points to the extra page.



## EMA SUBROUTINES

EMA is implemented by four subroutines:

- 1) EMAST
- 2) MMAP
- 3) .EMAP
- 4) .EMIO

## EMAST — RETURNS INFORMATION ABOUT EMA

CALL EMAST (NEMA, NMSEG, IMSEG)

where (all values are returned):

NEMA = total size of EMA

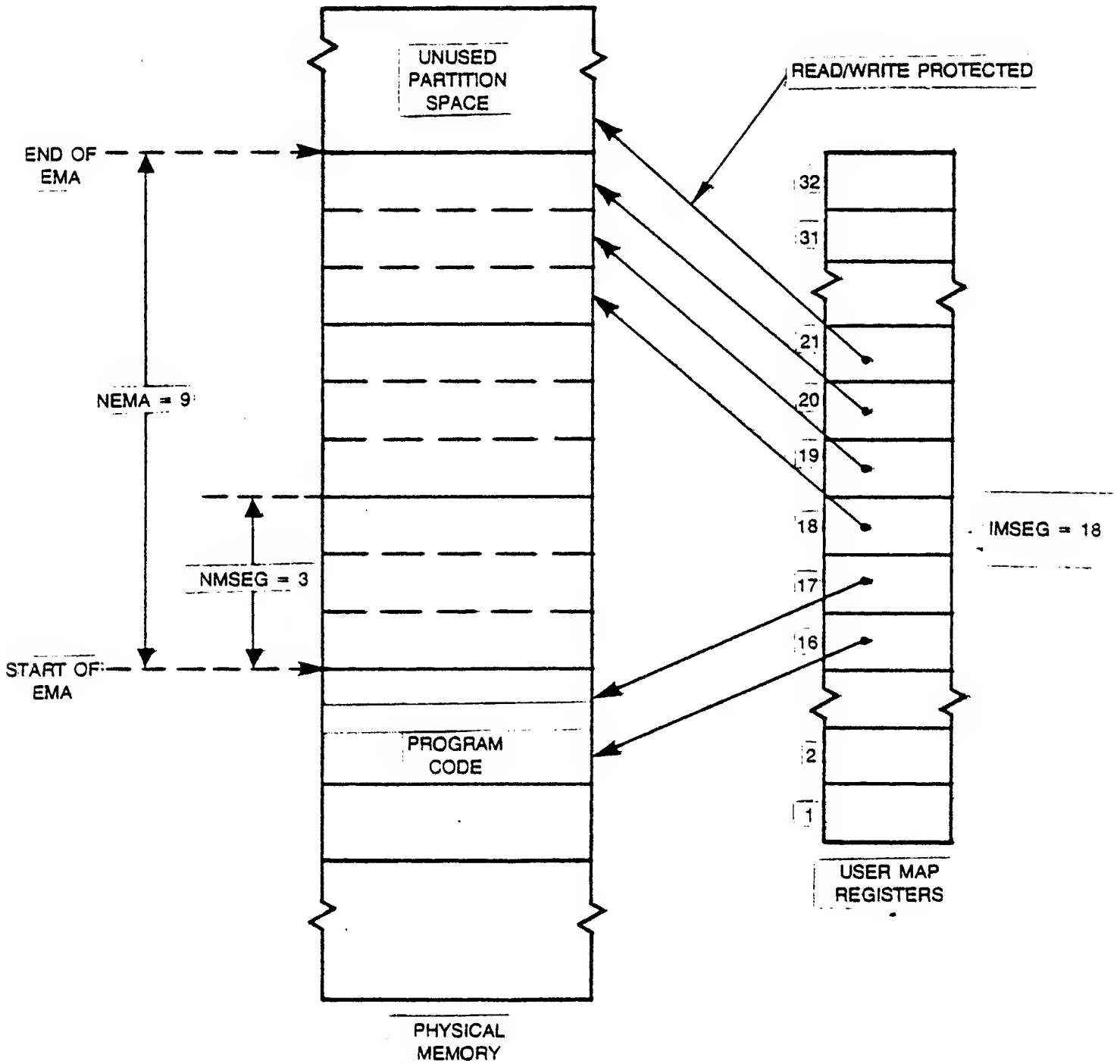
NMSEG = size of MSEG

IMSEG = starting logical page of EMA

JSB EMAST  
DEF RTN  
DEF NEMA  
DEF NMSEG  
DEF IMSEG  
RTN —

## EMAST EXAMPLE

	NAM	PROG
	ENT	PROG
XYZ	EMA	9,3



## MMAP - MAPS PHYSICAL PAGES INTO LOGICAL ADDRESS SPACE

```
CALL MMAP (IPGS,NPGS)

JSB MMAP
DEF RTN
DEF IPGS
DEF NPGS
RTN —
```

where:

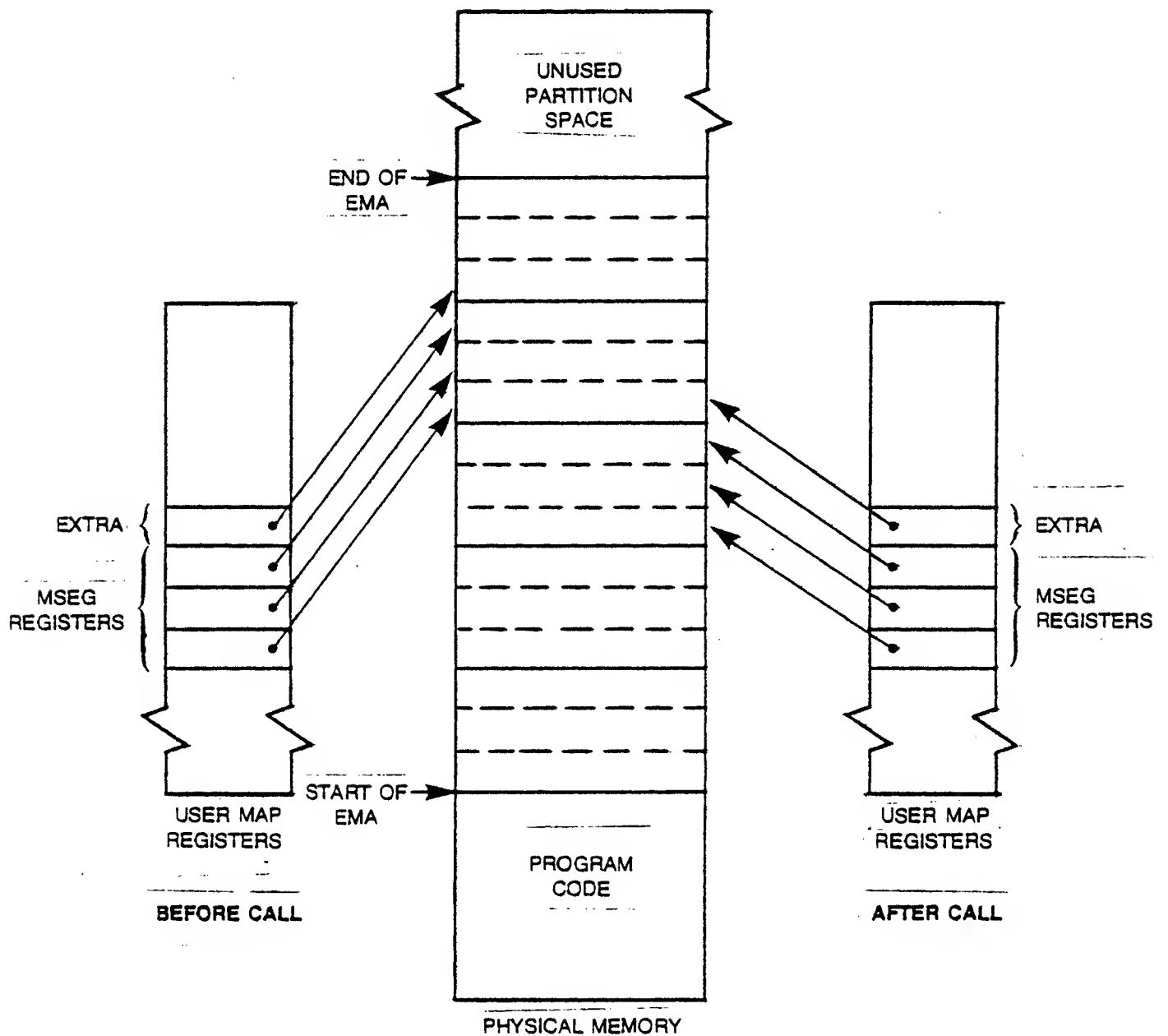
IPGS = page offset from start of EMA to segment being  
mapped (origin = 0)

NPGS = number of pages to map.

- MMAP maps one more page than the NPGS specified in case an element to be referenced "spills over" to the next page above the mapped portion.
- When mapping near the top of EMA, MMAP read/write protects pages above top of EMA.
- MMAP will not map more pages than the MSEG size.



## EXAMPLE MMAP CALL: CALL MMAP (6,3)



## TABLE DEFINITION FOR .EMIO AND .EMAP

- The Assembly Language programmer must build a table containing information about the EMA array to make calls to .EMIO and .EMAP.
- FTN4 compiler develops the table automatically.
- The form of the table is:

TABLE:	DEC	n	number of dimensions
	DEC	-L(n)	negative of lower bound of $n^{\text{th}}$ dimension
	DEC	d(n-1)	number of elements in (n-1) dimension
	DEC	-L(n-1)	
	DEC	d(n-2)	
	DEC	L(n-2)	
		.	
		.	
		.	
		.	
	DEC	d(2)	
	DEC	-L(2)	
	DEC	d(1)	
	DEC	-L(1)	
	DEC	p	number of words per element
	DEC	offset 1	(bits 15-0)
	DEC	offset 2	(bits 31-16)

where: L(i) is the lower bound of the  $i^{\text{th}}$  dimension.  
d(i) is the number of elements in the  $i^{\text{th}}$  dimension.  
offset 1 and 2 specifies the number of words between the start of EMA and this array.

## **.EMAP — RESOLVES REFERENCES TO EMA ELEMENTS**

- .EMAP maps the referenced EMA element into the program's logical address space and returns the element's logical address in the B-register.

JSB	.EMAP	
DEF	RTN	
DEF	ARRAY	name of start of EMA array
DEF	TABLE	table containing array parameters
DEF	$A_n$	actual subscript for $n^{\text{th}}$ dimension
DEF	$A_{n-1}$	actual subscript for $(n-1)^{\text{st}}$ dimension
	.	
	.	
	.	
DEF	$A_2$	actual subscript for 2nd dimension
DEF	$A_1$	actual subscript for 1st dimension
RTN	-error return-	A-reg = 15 (ASCII) B-reg = EM (ASCII)
	normal return	B-reg = logical address of element in current map

where: ARRAY is the name of the start of the EMA array.  
TABLE is as previously defined.

- FORTRAN compiler emits calls to .EMAP to resolve EMA references.

## .EMAP - SOFTWARE AND FIRMWARE DIFFERENCES

### SOFTWARE .EMAP:

Checks whether referenced element is already mapped.

YES - returns logical address of element in current map.

NO - maps in complete MSEG containing element, then returns logical address of element in new map.

### FIRMWARE .EMAP:

Always maps two pages, then returns logical address of element in new map. First page contains element; second page is mapped in case element "spills over" to next page.

EMALD T=00004 IS ON CR01001 USING 00003 BLKS R=0027

```

0001 ASMB,L,T
0002 *****
0003 * EXAMPLE ASSEMBLER PROGRAM, USING .EMAP, TO STORE *
0004 * 9000 CONSECUTIVE INTEGERS INTO AN EMA ARRAY OF SIZE
0005 * 9K, WITH AN MSEG SIZE OF 2K. *
0006 *****
0007     NAM EMALD
0008     EXT .EMAP
0009     EXT DBUGR
0010     EXT EXEC
0011 EMALB EMA 9,2           EMA SIZE=9 PAGES, MSEG SIZE=2 PAGES
0012 LOOP NOP               BEGIN EXECUTION
0013 *****
0014 * CALL .EMAP TO MAP ELEMENT AND RETURN PTR IN B-REG *
0015 *****
0016     JSB .EMAP           CALL .EMAP TO MAP ELEMENT
0017     DEF RTN             DEF RETURN ADDRESS
0018     DEF EMALB           DEF EMA AREA
0019     DEF TABLE          DEF TABLE FOR USE BY .EMAP
0020     DEF COUNT           DEF COUNTER FOR INDEX OF EMA ARRAY
0021 RTN JSB DBUGR          ERROR RETURNED
0022 *****
0023 * NORMAL RETURN= B-REG HOLDS PTR TO ELEMENT IN CURRENT *
0024 * LOGICAL ADDRESS SPACE *
0025 *****
0026     LDA COUNT          GET INDEX OF ARRAY
0027
0028     STA 1,1             STORE COUNT IN EMA ARRAY
0029     CPA TOP             CHECK TO SEE IF DONE
0030     JMP DONE            YES, EXIT
0031     ISZ COUNT          ADD ONE TO COUNT
0032     JMP LOOP           CONTINUE PROCESS WITH COUNT=COUNT+1
0033 *****
0034 * DATA STORAGE *
0035 *****
0036 COUNT DEC 1
0037 TOP DEC 9000           STORE 9000 ELEMENTS
0038 TABLE DEC 1           * OF DIMENSIONS IN ARRAY
0039     DEC -1             NEGATIVE OF LOWER BOUND OF FIRST DIM
0040     DEC 1               * WORDS ELEMENT
0041     DEC 0
0042     DEC 0
0043 DONE JSB EXEC          TERMINATE EXECUTION
0044     DEF *+2
0045     DEF D6
0046 D6 DEC 6              EXEC REQUEST CODE FOR TERMINATION
0047     END LOOP

```

## EXAMPLE PROGRAM FOR .EMAP AND MMAP

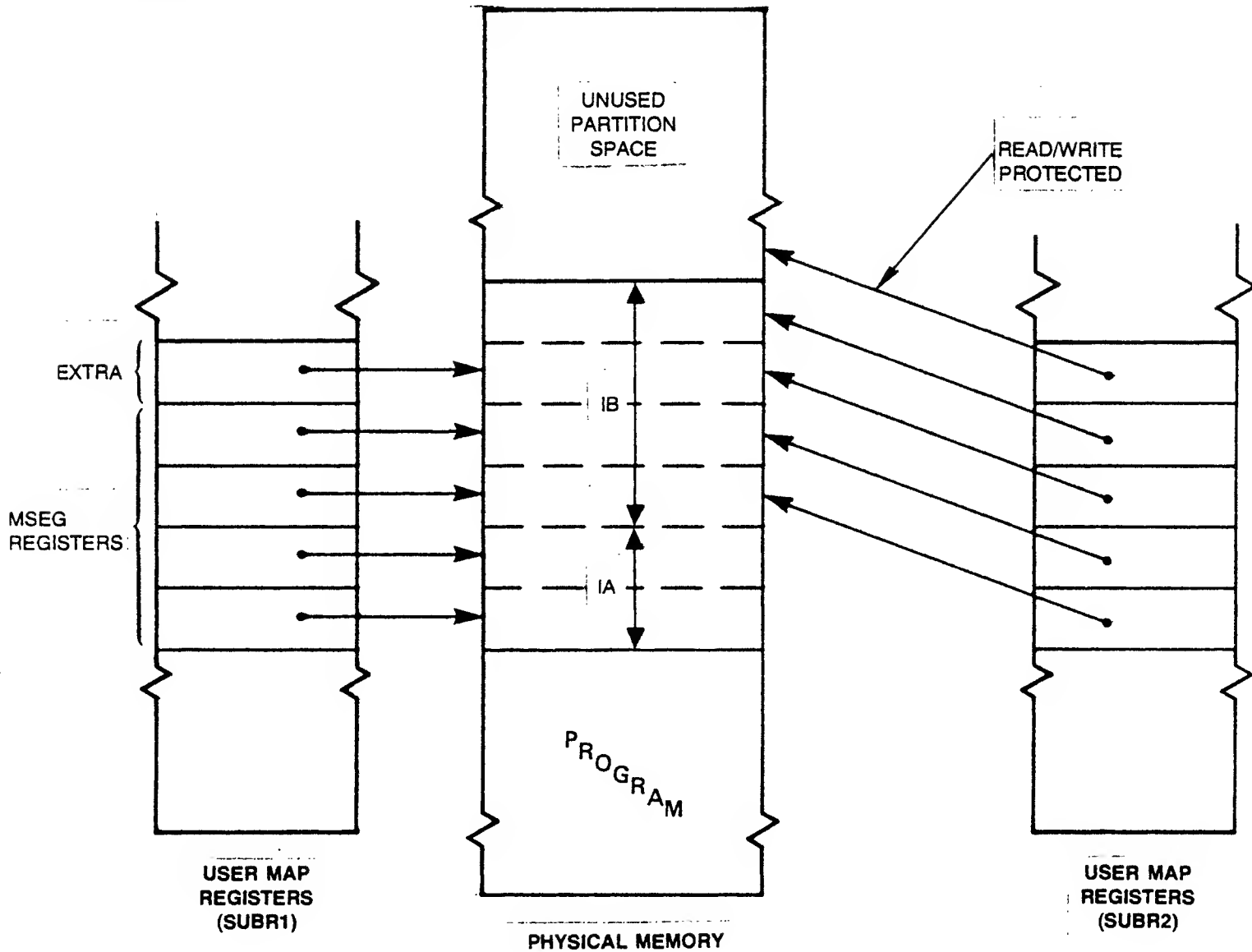
The following program illustrates the use of .EMAP and MMAP to load and manipulate EMA data.

SUBR1 and SUBR2 illustrate two different ways to access EMA variables. SUBR1 uses .EMAP to load 2048 consecutive values into IA. SUBR2 does its own mapping and uses the EMA label XYZ to load 4096 consecutive values into IB.

SUBR3 uses both techniques to add IA(2000) to IB(4000). Note especially the array table for IB in SUBR3 which contains an offset of 2048 (lines 60-61). This needs to be specified to reference IB since IB starts 2048 elements above the start of EMA.

## DIAGRAM FOR PROGRAM EMAP

EMA SIZE = 6 PAGES  
MSEG SIZE = 4 PAGES



```

C*****
C PROGRAM ILLUSTRATING THE USE OF .EMAP. THIS PROGRAM
C LOADS CONSECUTIVE VALUES INTO TWO EMA ARRAYS. ONE LOAD
C USES .EMAP AND THE OTHER LOAD USES THE EMA LABEL AND DOES
C ITS OWN CALL TO MMAP.
C*****
FTN4,L,T
$EMA(XYZ,4)
PROGRAM EMAP
COMMON/XYZ/IA(2048),IB(4096)
DIMENSION IPRAM(5)
C *****
CALL RMPAR(IPRAM)
LU=IPRAM(1)
IF (LU .LE. 0) LU=1
C *****
C LOAD CONSECUTIVE VALUES INTO IA USING .EMAP.
CALL SUBR1
DO 10 J=1,2048
IF (IA(J) .NE. J) CALL DBUGR
10 CONTINUE
C *****
C LOAD CONSECUTIVE VALUES INTO IB USING THE EMA LABEL
CALL SUBR2
WRITE(LU,900) (IB(K),K=1,10)
900 FORMAT(" EMA OUTPUT"// " FIRST 10 VALUES OF IB=",10I4)
DO 20 K=1,4096
IF (IB(K) .NE. K) CALL DBUGR
20 CONTINUE
C *****
C SUBR3 CALLS MMAP AND .EMAP TO ADD IA(2000) TO IB(4000)
ITEMP=IA(2000)+IB(4000)
WRITE(LU,2000) ITEMPS
2000 FORMAT("/" BEFORE SUBR3,IA(2000)+IB(4000)=" ,I6)
CALL SUBP3
WRITE(LU,3000) IB(4000)
3000 FORMAT(" AFTER SUBR3,IB(4000)=" ,I6)
IF (IB(4000) .NE. ITEMPS) CALL DBUGR
C *****
WRITE(LU,1000)
1000 FORMAT("//" /EMAP:ENDING NOW"//)
END

```



```

ASMB,R,L
*****
* SUBR1 LOADS 2048 CONSECUTIVE VALUES INTO THE ARRAY NAMED
* IA. IA'S ARRAY TABLE IS LISTED AT TABLE.
*****
    NAM SUBR1,7
    ENT SUBR1
    EXT XYZ
    EXT .EMAP
    EXT DBUGR
SUBR1 NOP
LOOP NOP
*****
* CALL .EMAP TO MAP ELEMENT AND RETURN PTR IN B-REG
*****
    JSB .EMAP
    DEF RTN1
    DEF XYZ
    DEF TABLE
    DEF COUNT
RTN1 JSB DBUGR
*****
* NORMAL RETURN- B-REG HOLDS PTR TO ELEMENT IN CURRENT
* LOGICAL ADDRESS SPACE.
*****
    LDA COUNT
    STA 1,I
    CPA TOP
    JMP DONE
    ISZ COUNT
    JMP LOOP
DONE NOP
    JMP SUBR1,I
*****
* DATA STORAGE STARTS HERE
*****
COUNT DEC 1
TOP DEC 2048
TABLE DEC 1          # OF DIMENSIONS IN ARRAY
    DEC -1          NEGATIVE OF LOWER BOUND OF FIRST DIM
    DEC 1           # WORDS/ELEMENT
    DEC 0
    DEC 0
    END

```

```

ASMB,L,R
*****
* SUBR2 LOADS 4096 CONSECUTIVE INTEGERS INTO ARRAY IB, USING
* THE EMA LABEL OF XYZ.
*****
    NAM SUBR2,7
    ENT SUBR2
    EXT XYZ
    EXT DBUGR
    EXT MMAP
ADXYZ DEF XYZ
SUBR2 NOP
*****
* CALL MMAP TO MAP IN EMA PAGES CORRESPONDING TO IB ARRAY
    JSB MMAP
    DEF PTN1
    DEF IPGS
    DEF NPGS
PTN1 CPA MIN1
    JSB DBUGR
*****
* LOAD PTR TO EMA INTO B-REG
    LDB ADXYZ
    RBL,CLE,SLB,ERB GET RID OF SIGN BIT
    LDB 1,I          RESOLVE ONE INDIRECT
*****
* LOAD COUNT INTO A-REG, STORE IT THROUGH THE B-REG
LOOP LDA COUNT
    STA 1,I
    CPA TOP
    JMP DONE
*****
* BUMP B-REG, BUMP COUNT, LOOP BACK **
    ADB D1
    ISZ COUNT
    JMP LOOP
*****
* RETURN
DONE JMP SUBR2,I
*****
* DATA VALUES START HERE
D1    DEC 1
MIN1  DEC -1
COUNT DEC 1          COUNTS HOW MANY VALUES LOADED
TOP    DEC 4096        TOP VALUE LOADED
IPGS   DEC 2          PAGE OFFSET FOR MMAP CALL
NPGS   DEC 4          # OF PAGES MAPPED IN CALL TO MMAP
END

```

```

ASME,L,R
*****
* SUBR3 ADDS IA(2000) TO IB(4000) AND STORES THE RESULT IN
* IB(4000)
*****
    NAM SUBR3,7
    ENT SUBR3
    EXT XYZ
    EXT .EMAP
    EXT MMAP
    EXT DBUGR
ADXYZ DFF XYZ
SUBR3 NOP
*****
* GET IA(2000) USING MMAP TO DO YOUR OWN MAPPING
    JSB MMAP
    DEF PTN1
    DEF IPGS
    DEF NPGS
PTN1  CPA MIN1
    JSB DBUGR
*****
* LOAD A PTR TO IA(2000) INTO B-REG
    LDB ADXYZ
    RBL,CLE,SLB,ERB
    LDB 1,I
    ADB D1999      ADD IN 1999 OFFSET TO GET IA(2000)
*****
* PUT IA(2000) INTO A-REG,STORE IT AT ARG1
    LDA 1,I
    STA ARG1
*****
* USE .EMAP TO GET PTR TO IB(4000)
    JSB .EMAP
    DEF RTN2
    DEF XYZ
    DEF TBL2
    DEF D4000
RTN2  JSB DBUGR
*****
* NORMAL RTN, B-REG PTS TO IB(4000)
* LOAD THE SECOND ARG INTO A-REG.
    LDA 1,I
*****
* ADD IN ARG1 AND STORE BACK INTO IB(4000)
    ADA ARG1
    STA 1,I
    JMP SUBR3,I
*****
* DATA STORAGE STARTS HERE.
ARG1  NOP          VALUE OF IA(2000) STORED HERE
IPGS  DEC 0        MMAP MAPS STARTING AT 0 OFFSET
NPGS  DEC 4        MMAP MAPS 4 PAGES
MIN1  DEC -1       CPECKS ERROR RETURN FOR MMAP CALL
D1999 DEC 1999     OFFSET INTO ARRAY
D4000 DEC 4000     ELEMENT # FOR .EMAP CALL
TBL2  DEC 1        # OF DIMENSIONS IN ARRAY
      DEC -1       NEG OF LOWER BOUND OF FIRST DIMENSION
      DEC 1        # WORDS/ELEMENT
      DEC 2048     OFFSET FROM START OF EMA FOR THIS ARRAY
      DEC 0        HIGH BITS OF OFFSET
      END

```

EMAP OUTPUT

FIRST 10 VALUES OF IH= 1 2 3 4 5 6 7 8 9 10

BEFORE SUBR3, IA(2000)+IB(4000)= 6000

AFTER SUBR3, IB(4000)= 6000

/EMAP:ENDING NOW

## `.EMIO` - USED FOR I/O FROM EMA ARRAYS

- `.EMIO` - used to ensure that an entire memory buffer is in a program's logical address space (possibly for I/O purposes).
- Only callable in Assembly language.
- Allows EXEC I/O and FMP calls with EMA buffers.
- `.EMIO` does the following:
  - 1) Checks if buffer fits into any possible mapping  
YES - Go to Step 2  
NO - Error Return
  - 2) Check if buffer fits into standard MSEG  
YES - map in the MSEG, return logical address of buffer.  
NO - go to Step 3
  - 3) Map in a non-standard MSEG and return logical address of buffer in the non-standard MSEG.
- The user buffer plus a possible page offset of the start of the buffer should be less than the MSEG size. A good rule of thumb is to always make sure that the buffer is at least one page smaller than the MSEG size.
- `.EMIO` always maps an MSEG size number of pages.

## .EMIO CALLING SEQUENCE

- .EMIO callable only from Assembly Language.

EXT .EMIO

⋮

JSB .EMIO

DEF RTN

DEF BUFL                      LENGTH OF BUFFER

DEF TABLE                    TABLE DESCRIBING EMA ARRAY

DEF a(n)

DEF a(n-1)

⋮

⋮

DEF a(2)

DEF a(1)

} ARRAY SUBSCRIPT VALUES FOR START  
OF BUFFER

RTN - error return - A-reg = 16 (ASCII) B-reg = EM(ASCII)  
                  normal return    B-reg = logical address of the element  
  in current map.  
                  A-reg = meaningless.

## EXAMPLE PROGRAM FOR .EMIO CALL

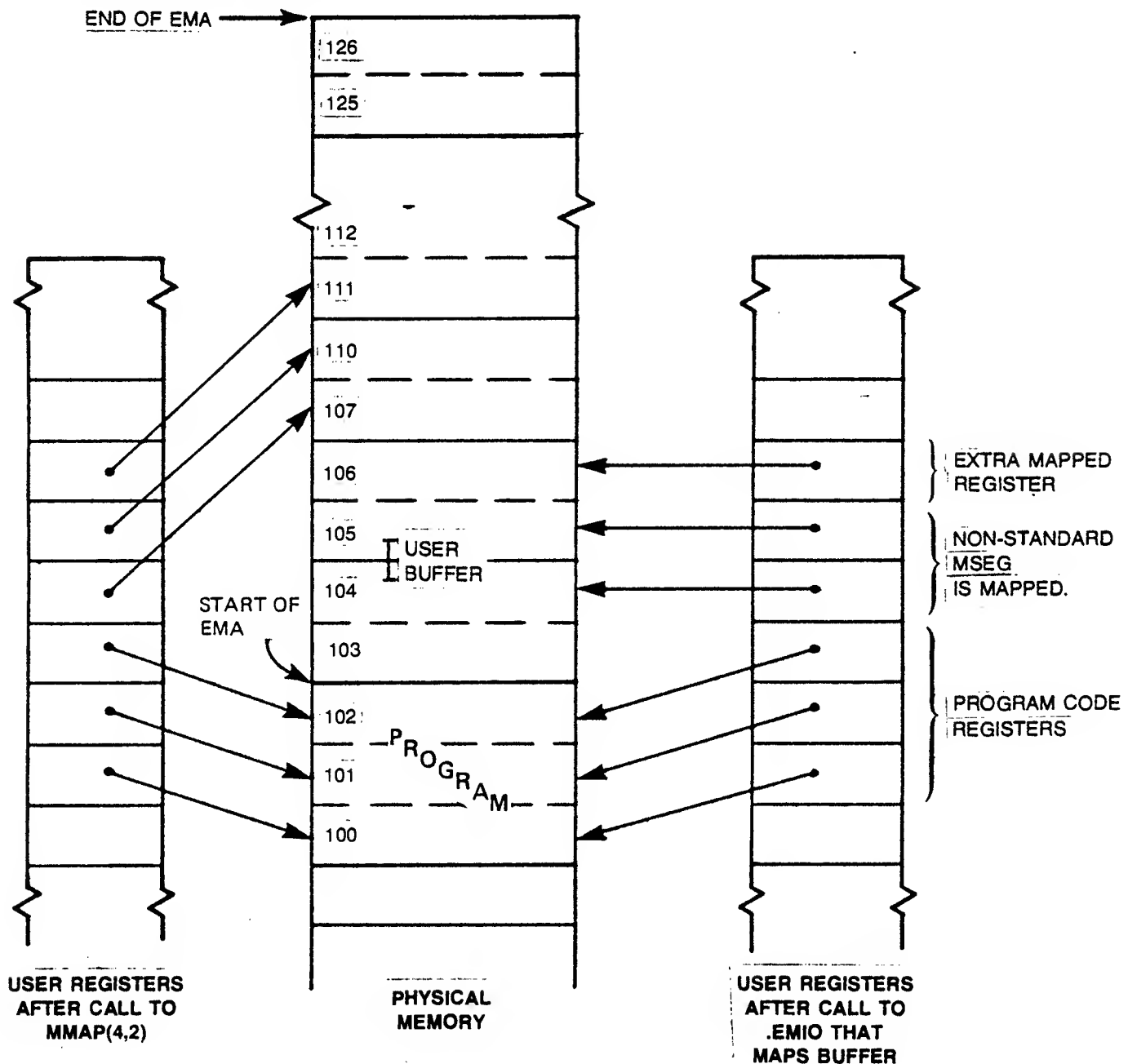
The following example program demonstrates the use of .EMIO to output a buffer that crosses an MSEG boundary. First it loads the buffer with ASCII data (lines 20 to 26), puts out some explanatory material (lines 29-42), then calls SUBR to output the buffer.

SUBR calls .EMIO to map in a non-standard MSEG, then calls EXEC to output the desired buffer. After SUBR returns, program EMIO outputs a listing of the user registers (lines 50-52).

Note especially registers 14 and 15 in the printed output that reflect the non-standard MSEG that was mapped in.

# **DIAGRAM EXPLAINING .EMIO EXAMPLE PROGRAM**

**REGISTERS 14 AND 15 POINT TO EMA AREA; USER WANTS TO OUTPUT BUFFER THAT CROSSES MSEG BOUNDARY.**



Note: One extra register is always mapped in case an element overflows to the next page.



```

C*****
C PROGRAM TO SHOW USE OF .EMIO. THIS PROGRAM CALLS A
C SUBR THAT CALLS .EMIO TO MAP IN A BUFFER THAT CROSSES
C AN MSEG BOUNDARY.
C*****
FTN4,L
SEMA(XYZ,2)
    PROGRAM EMIO
    COMMON/XYZ/IA(20000)
    DIMENSION IMAP(32),IPRAM(5)
C *****
    CALL RMPAR(IPRAM)
    LU=IPRAM(1)
    IF (LU .LE. 0) LU=1
    WRITE(LU,1400)
1400  FORMAT(///" OUTPUT FROM EMIO DEMO PROGRAM"//)
C *****
C PUT MESSAGE INTO BUFFER
C NOTE:MESSAGE CROSSES AN MSEG BOUNDARY
    DO 20 K=2038,2058,5
        IA(K)=2HEM
        IA(K+1)=2HIO
        IA(K+2)=2HWO
        IA(K+3)=2HRK
        IA(K+4)=2HS!
20    CONTINUE
C *****
C IMSEG=FIRST REGISTER THAT POINTS TO EMA
    CALL EMAS(TNEMA,NMSEG,IMSEG)
    WRITE(LU,1500) IMSEG
1500  FORMAT("/" /EMIO: THIS REG POINTS TO FIRST PAGE MSEG",I6/)
C *****
C MAP TWO PAGES OF EMA STARTING AT OFFSET OF 4 PAGES
    CALL MMAP(4,2)
C *****
C IMAP IS A COPY OF THE USER MAP REGISTERS
    WRITE(LU,1600)
1600  FORMAT("/" THE 32 USER REGISTERS ARE:")
    CALL EXEC(26,IFPG,ILMEM,NPGS,IMAP)
    WRITE(LU,1000) (IMAP(J),J=1,32)
1000  FORMAT(1007)
    WRITE(LU,9000)
C *****
C CALL SUBR THAT CALLS .EMIO AND OUTPUTS A BUFFER
C THAT CROSSES AN MSEG BOUNDARY.
    CALL SUBR
    WRITE(LU,9000)
C *****
C CALL EXEC TO GET INFO ON USER MAP AGAIN
    CALL EXEC(26,IFPG,ILMEM,NPGS,IMAP)
    WRITE(LU,2000) (IMAP(J),J=1,32)
2000  FORMAT(1007)
C *****
C FINISH UP THE PROGRAM
    WRITE(LU,3000)
3000  FORMAT(///" ENDING PROGRAM EMIO"/)
9000  FORMAT(///)
    END

```

```

ASMB,R,L
*****
* THIS SUBR CALLS .EMIO TO MAP IN A NONSTANDARD MSEG
* THAT CROSSES AN MSEG BOUNDARY. THEN IT CALLS EXEC
* TO OUTPUT A BUFFER THAT CROSSES THE MSEG BOUNDARY
*****
    NAM SUBR,7
    ENT SUBR
    EXT .EMIO
    EXT ERROR
    EXT EXEC
SUBR  NOP
*****
* SET UP MSEG TO POINT TO ELEMENTS 2038-2058 IN EMA
*****
    JSB .EMIO
    DEF RTN1
    DEF BUFL
    DEF TABLE
    DEF A1
RTN1  JSB EPROR
*****
* NORMAL RETURN, B-REG HOLDS LOGICAL ADDRESS OF ELEMENT A1
    STB ADDR
*****
* JSB EXEC TO OUTPUT BUFFER TO LU 6
    JSB EXEC
    DEF RTN2
    DEF ICODE
    DEF ICNWD
    DEF ADDR,I
    DEF BUFL
RTN2  NOP
*****
* RETURN POINT
    JMP SUBR,I
*****
* PARAMETERS FOR CALLS TO .EMIO AND EXEC
BUFL  DEC 20          BUFFER LENGTH
A1    DEC 2038        INDEX OF ARRAY ELT
ICODE DEC 2           EXEC WRITE ICODE
ICNWD OCT 206         OUTPUT TO LU 6,PRINT COLUMN ONE
*****
* TABLE FOR CALL TO .EMIO
TABLE DEC 1           # OF DIMENSIONS
      DEC -1          NEGATIVE OF LOWER BOUND OF FIRST DIMENSION
      DEC 1           # OF WORDS PER ELT
      DEC 0           OFFSET WORD #1
      DEC 0           OFFSET WORD #2
*****
* ADDR HOLDS LOGICAL ADDRESS OF DESIRED ELT AFTER CALL
* TO .EMIO AND STB INSTRUCTION
ADDR  DEC 0
*****
    END SUBR

```

# OUTPUT FROM EMIO DEMO PROGRAM

/EMIO: THIS REG POINTS TO FIRST PAGE MSEG 14

THE 32 USER REGISTERS ARE:

000077	000001	000002	000003	000004	040005	040006	040007	040010	040011
040012	000100	000101	000102	000107	000110	000111	140001	140002	140003
140004	140005	140006	140007	140010	140011	140012	140013	140014	140015
140016	140017								

EMIOWORKS!EMIOWORKS!EMIOWORKS!EMIOWORKS!

000077	000001	000002	000003	000004	040005	040006	040007	040010	040011
040012	000100	000101	000102	000104	000105	000106	140001	140002	140003
140004	140005	140006	140007	140010	140011	140012	140013	140014	140015
140016	140017								

ENDING PROGRAM EMIO

EMA Registers → Extra Register

FUNCTION	NUMBER OF SUBSCRIPTS							
	STANDARD MEMORY ( $\mu$ s)				HIGH SPEED MEMORY ( $\mu$ s)			
	0D	1D	2D	3D	0D	1D	2D	3D
21MX Software								
.EMAP no map	326	429	529	629				
map	1,080+	1,182+	1,283+	1,384+				
	16.8M	16.8M	16.8M	16.8M				
.EMIO no map	321	424	525	624				
map	1,083+	1,186+	1,288+	1,386+				
	16M	16M	16M	16M				
21MX-E $\mu$ Code								
.EMAP	35	45	55	65	32	40	49	58
.EMIO no map	44	54	64	74	40	49	57	66
map	87+1.3M	96+1.3M	106+1.3M	115+1.3M	82+1.2M	92+1.2M	98+1.2M	109+1.2M
M = #MSEG								

## APPENDIX A



## SYSTEM TABLES

- ID segments, long, short, & extensions
- Equipment Table
- Device Reference Table
- Interrupt Table
- Track Assignment Table
- Class Table
- LU Switch Table
- Resource Number Table
- Keyword Block
- ID Extension Table
- Memory Allocation Table
- Memory Protect Fence Table
- Driver Mapping Table
- Track Map Table

## SYSTEM LISTS

- Schedule List
- General Wait List
- Available Memory Suspend List
- Disc Allocation Suspend List
- Operator Suspend List
- I/O Suspend Lists
- Free SAM List

# SYSTEM BASE PAGE COMMUNICATION AREA

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
01645	XIDEX	Address of current program's ID extension
01646	XMATA	Address of current program's MAT entry
01647	XI	Address of index register save area
01650	EQTA	FWA of Equipment Table
01651	EQT#	Number of EQT entries
01652	DRT	FWA of Device Reference Table, word 1
01653	LUMAX	Number of logical units in DRT
01654	INTBA	Fwa of Interrupt Table
01655	INTLG	Number of Interrupt Table Entries
01656	TAT	FWA of Track Assignment Table
01657	KEYWD	FWA of keyword block
I/O MODULE/DRIVER COMMUNICATION		
01660	EQT1 \	Addresses of first 11 words of current EQT entry (see 01771 for last four words)
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EQT5 \	
01665	EQT6 /	
01666	EQT7	
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11 /	
01673	CHAN	Current DCPC channel number
01674	TEG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY
SYSTEM REQUEST PROCESSOR/EXEC COMMUNICATION		
01676	RQCNT	Number of request parameters -1
01677	RQPTN	Return point address
01700	RQP1 \	Addresses of request parameters (set for a maximum of nine parameters)
01701	RQP2	
01702	RQP3	
01703	RQP4 \	
01704	RQP5 /	
01705	RQP6	
01706	RQP7	
01707	RQP8	
01710	RQP9 /	



# SYSTEM BASE PAGE COMMUNICATION AREA (continued)

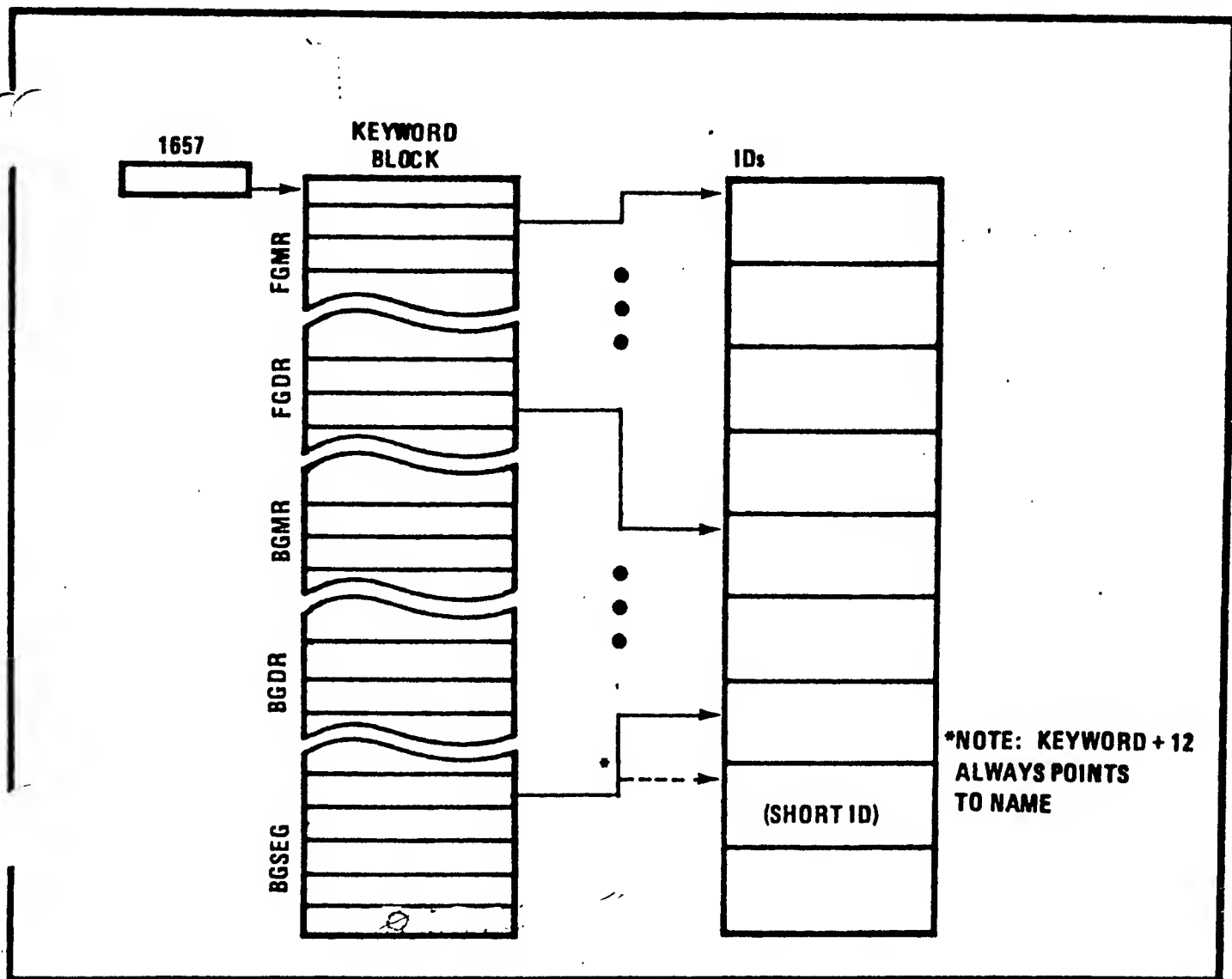
Octal Location	Contents	Description
SYSTEM LISTS ADDRESSES		
01711	SKEDD	Schedule list
01713	SUSP2	Wait Suspend list
01714	SUSP3	Available Memory list
01715	SUSP4	Disc Allocation list
01716	SUSP5	Operator Suspend list
PROGRAM ID SEGMENT DEFINITION		
01717	XECT	ID segment address of current program
01720	XLINK	Linkage
01721	XTEMP	Temporary (five words)
01726	XPRI0	Priority word
01727	XPENT	Primary entry point
01730	XSUSP	Point of suspension
01731	XA	A-register at suspension
01732	XB	B-register at suspension
01733	XEO	E and overflow register suspension
SYSTEM MODULE COMMUNICATION FLAGS		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy interface flag
01740	IDSDA	Disc address of first ID segment
01741	IDSDP	Position within disc sector
MEMORY ALLOCATION BASES DEFINITION		
01742	BPA1	FWA user base page link area
01743	BPA2	LWA user base page link area
01744	BPA3	FWA user base page link
01745	LBORG	FWA of resident library area
01746	RTORG	FWA of real-time COMMON
01747	RTCOM	Length of real-time COMMON
01750 D	RTDRA	FWA of real-time partition
01751 D	AVMEM	LWA+1 of real-time partition
01752	LGCRG	FWA of background COMMON
01753	BGCOM	Length of background COMMON
01754 D	BGDRA	FWA of background partition

SYSTEM BASE PAGE COMMUNICATION AREA (continued)

Octal Location	Contents	Description
UTILITY PARAMETERS		
01755	TATLG	Negative length of track assignment table
01756	TATSD	Number of tracks on system disc
01757	SECT2	Number of sectors/track on LU2 (system)
01760	SECT3	Number of sectors/track on LU3 (aux.)
01761	DSCLB	Disc address of user available library entry points
01762	DSCLN	Number of user available library entry points.
01763	SYSLB	Disc address of system library entry points
01764	SYSLN	Number of system library entry points
01765	LGOTK	LGO: LU#, starting track, number of tracks (same format as ID segment word 28)
01766	LGOC	Current LGO track/sector address (same format as ID segment word 26)
01767	SFCUN	LS: LU# and disc address (same format as ID segment word 26)
01770	MPTFL	Memory protect ON/OFF flag (0/1)
01771	EQT12 \	Address of last four words of current EQT
01772	EQT13 \	
01773	EQT14 /	
01774	EQT15 /	
01775 D	FENCE	Memory protect fence address
01777	BGLWA	LWA memory background partition
D letter indicates the contents of the location are set dynamically by the dispatcher.		

BASE PAGE EXAMPLE

```
LOCATIONS      1647 THROUGH          1746  
0036306 002023 000023 002762 000051 003104 000072 023717★<           ) D :'  
016071 002136 002137 002140 002141 002142 002143 002144★ 9 ↑ +  
002145 002146 002147 002150 000006 000015 002042 000003★                "  
051474 000002 047337 050534 051534 000000 000000 000000★9<   N Q\8\  
000000 000000 016535 000000 016406 000000 000000 000000★               ]  
016535 016535 016536 016537 016540 016541 016542 016543★ J J ↑ +  
016544 016545 016546 016547 016550 000000 000000 031017★                   2  
000000 000224 000057 000002 001445 000002 026000 011107★       /    %    ,    G  
  
LOCATIONS      1747 THROUGH          1777  
000144 052654 052654 011253 000525 052654 177400 000400★     U U         UU  
000140 000000 010422 001224 004646 000234 000000 000000★  
000000 000000 002151 002152 002153 002154 030000 000000★                 W  
052654                               ★U
```



#### I.D. definition:

Location 1657B on base page specifies the first entry in the keyword block. The keyword block in turn contains 1-word entries, each pointing to an ID segment\*, Last entry = 0. Keyword Block entries are ordered at generation by program type.

The keyword block entry (ID address) + 12 always points to the name-word. Thus, keyword entries for short ID's don't point to the first word of the ID.

## ID SEGMENT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																			
List Linkage																Word	0	\	
-----																			
TEMP 1																	1		
TEMP 2																	2		
TEMP 3																	3		
TEMP 4																	4		
TEMP 5																	5		
-----																			
Priority																	6		
Primary Entry Point																	7	*	
-----																			
Point of Suspension																	8		
A-Register																	9		
B-Register																	10		
EO-Registers																	11		
-----																			
Name 1								Name 2								12	*	\	Memory Resident Programs
Name 2								Name 4								13	*	/	
-----																			
Name 3								TM ML // SS Type								14	*		
-----																			
NA // NP W A // O // R D // // Status																	15		
-----																			
Time List Linkage																	16		
-----																			
RES   T Multiple																	17		
-----																			
Low Order 16 Bits of Time																	18		
-----																			
High Order 16 Bits of Time																	19		
-----																			
BA FW M AT RM RE PW RN Father ID Segment No.																	20		
-----																			
RP #pgs. (no BP)   MPFI   // Partition No. -1																	21		
-----																			
Low Main Address																	22	*	
-----																			
High Main Address + 1																	23	*	
-----																			
Low Base Page Address																	24	*	
-----																			
High Base Page Address + 1																	25	*/	
-----																			
LU Program: Track								Sector								26	*		
-----																			
LU Swap: Track								No. Tracks								27			
-----																			
ID Extension No.								EMA Size								28			
-----																			
High Address + 1 of Largest Segment																	29		
-----																			
Reserved																	30	\	
-----																			
Reserved																	31	\	
-----																			
Negative MTM LU number																	32	/	
-----																			

where:

\* = words used in short ID segments for program segments

TM = temporary load (copy of ID segment is not on the disc)

ML = memory lock (program may not be swapped)

SS = short segment (indicates a nine-word ID segment)

Type = specified program type (1-5)

NA = no abort (instead, pass abort errors to program)

NP = no parameters allowed on reschedule

W = wait bit (waiting for program whose ID segment address is in word 2)

A = abort on next list entry for this program

O = operator suspend on next schedule attempt

R = resource save (save resources when setting dormant)

D = dormant bit (set dormant on next schedule attempt)

Status = current program status

T = time list entry bit (program is in the time list)

BA = batch (program is running under batch)

FW = father is waiting (father scheduled with wait)

M = Multi-Terminal Monitor bit

AT = attention bit (operator has requested attention)

RM = reentrant memory must be moved before dispatching program

RE = reentrant routine now has control

PW = program wait (some other program wants to schedule this one)

RN = Resource Number either owned or locked by this program

RP = reserved partition (only for programs that request it)

MPFI = memory protect fence index

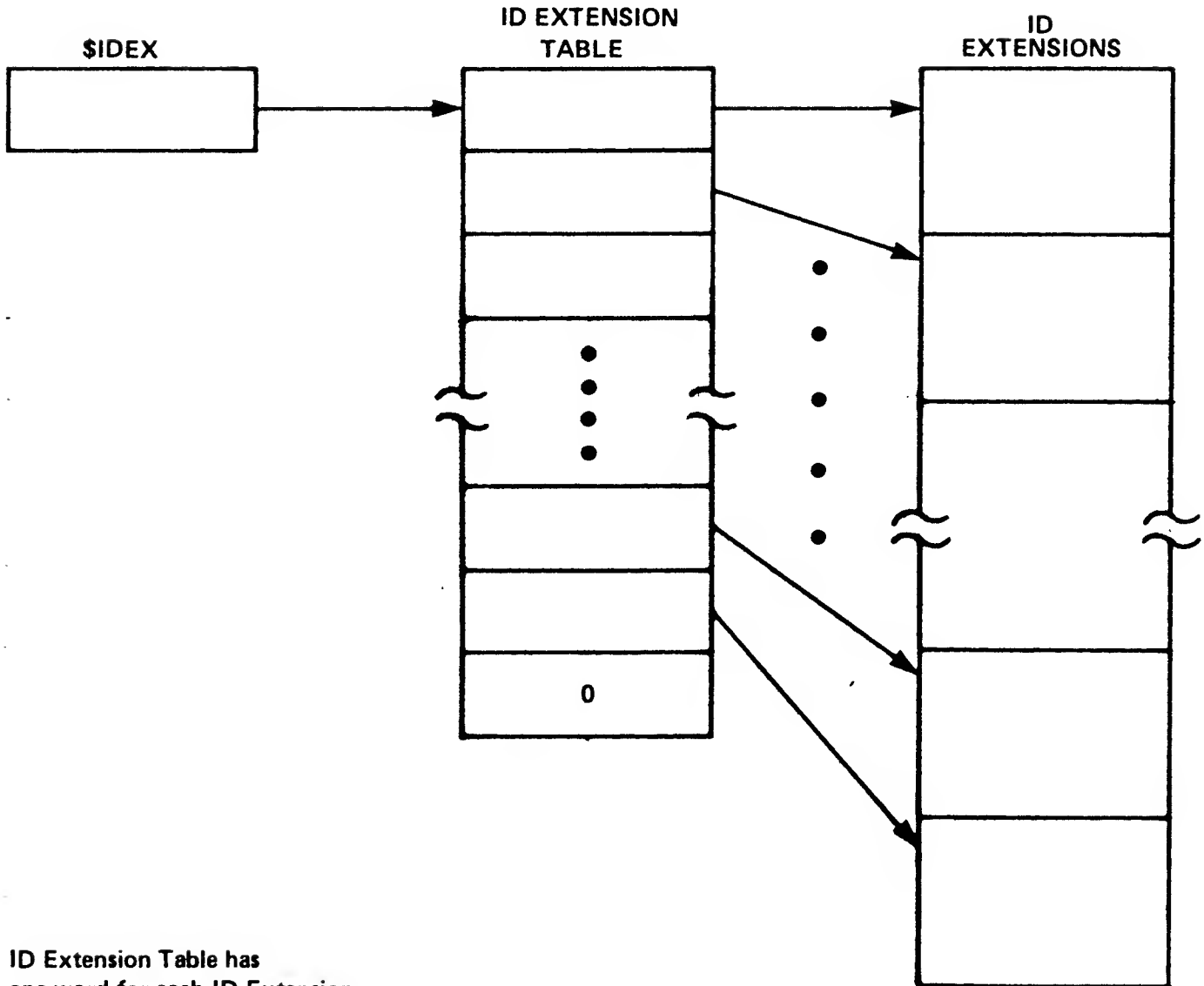
# LONG ID SEGMENT

ID	SEG OF	LUMAP			
WORD	LOCATION	VALUE(8)	VALUE(16)	VALUE(AS)	
1	17202	17551	8041		
2	17203	15	13		
3	17204	103750	-30744		
4	17205	177776	-	2	
5	17206	0	0		
6	17207	0	0		
7	17210	24	20		
8	17211	26120	11344		,P
9	17212	26221	11409		,
10	17213	0	0		
11	17214	17203	7811		
12	17215	126214	-21364		
13	17216	46125	19541		LU
14	17217	46501	19777		MA
15	17220	50003	20483		P
16	17221	0	0		
17	17222	0	0		
18	17223	0	0		
19	17224	25030	10752		*
20	17225	177574	- 132		
21	17226	0	0		
22	17227	11205	4741		
23	17230	26000	11264		,
24	17231	34356	14382		8.
25	17232	2	2		
26	17233	23	19		
27	17234	4122	2130		P
28	17235	0	0		
29	17236	0	0		
30	17237	0	0		
31	17240	0	0		
32	17241	0	0		
33	17242	177777	- 1		

# SHORT ID SEGMENT

ID	SEG OF	INDEX			
WORD	LOCATION	VALUE(8)	VALUE(16)	VALUE(AS)	
1	22650	32770	13816		5
2	22654	43115	17997		FM
3	22655	43522	18258		GR
4	22656	30023	12309		0
5	22657	32770	13816		5
6	22660	41032	16922		B
7	22661	45	37		%
8	22662	105	69		E
9	22663	2714	1484		

# ID EXTENSION TABLE



ID Extension Table has  
one word for each ID Extension.



# ID EXTENSION

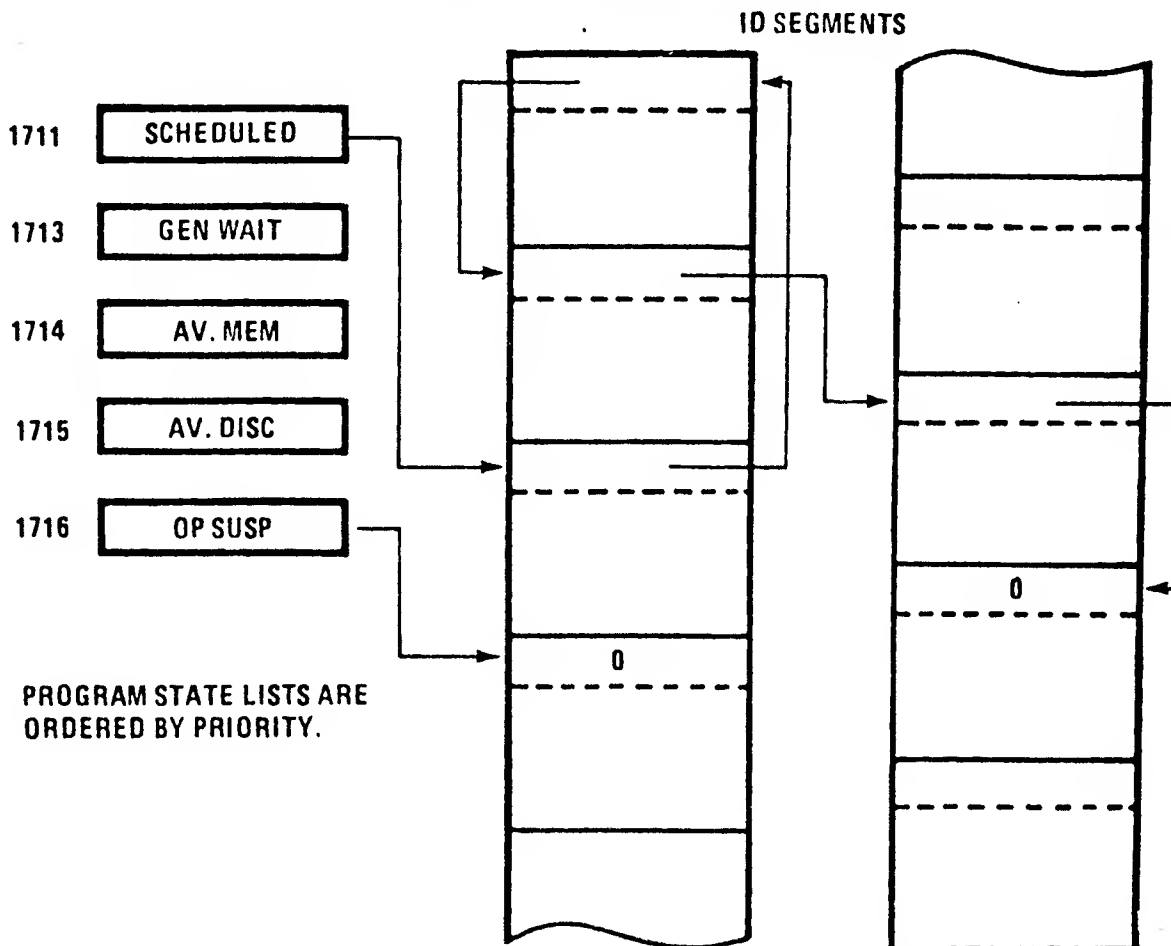
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																
NS		Current MSEG No.									# Pages MSEG					Word 0
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																
MSEG Start					DE	(Physical) EMA Start										Word 1
Page (logic.)						Page										
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																
////////////////////						# Tracks for EMA Swap										Word 2
////////////////////																
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																

where:

- NS = 0 if the MSEG is pointing to a standard segment of the EMA (set up by .EMAP)
- = 1 if the MSEG is pointing to a non-standard segment (set up by .EMIO or .EMAP) -
- DE = 0 if the EMA size was specified by the user
- = 1 if the EMA size is allowed to default to the maximum size available to the system

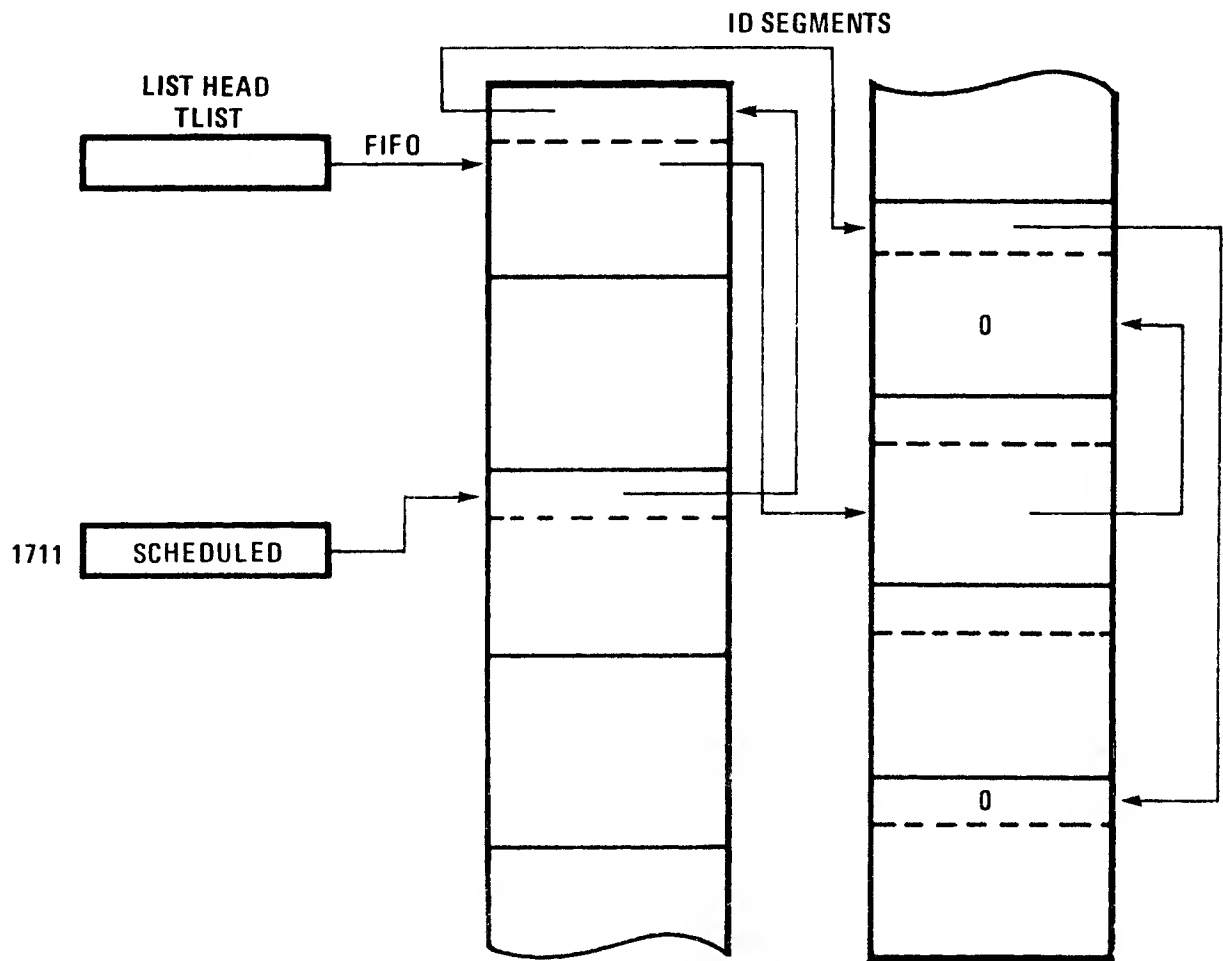
A program's ID segment word 28 contains its ID extension number.

# PROGRAM STATES LIST LINKING



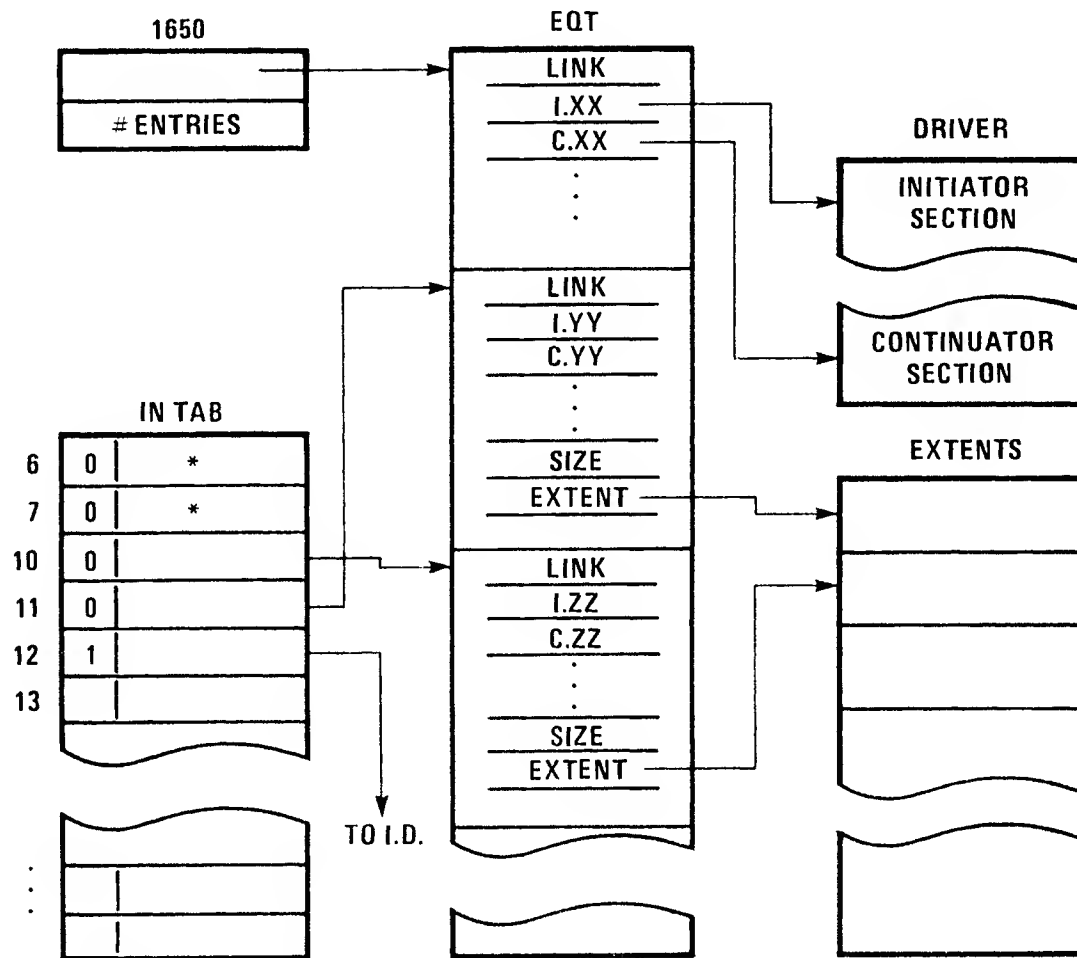
A linked list of ID segments is maintained for each of the major program states (accept I/O, to be seen later). The lists are prioritized and have their heads on Base Page. ID link word is word #1.

# TIME LIST



The time list's head is at TLIST in RTIME, and is linked thru ID word 17. It threads independently of the state lists, and is FIFO (vs prioritized).

# EQT LINKING



RTEU/P-57

\* INTAB entries 6 and 7 are dynamic, reflecting the assignment of DMA channels to various EQT's as needed.

+ = EQT  
 - = PRG/ENT  
 0 = not used

# EQUIPMENT TABLE:

WORD	CONTENTS
1	* I/O LIST . LINK POINTER *
2	*DRIVER *INITIATION ADDRESS*
3	*DRIVER *COMPLETION ADDRESS*
4	*DBPOT/----UNIT#--CHANNEL #*
5	*AV-TYPE CODE- UNIT STATUS*
6	*REQUEST CONTROL WORD *
7	*REQUEST BUFFER ADDRESS *
8	*REQUEST BUFFER LENGTH *
9	*TEMPORARY OR DISC TRACK # *
10	*TEMPORARY OR DISC SECTOR #*
11	*DRIVER TEMPORARY STORAGE*
12	* " " " " (EXT. SIZE)
13	* " " " " (EXT. ADR.)
14	* DEVICE CLOCK RESET VALUE *
15	* " " " WORKING " *

D: =1 IF A DMA CHANNEL REQUIRED FOR TRANSFER

B: =1 IF AUTOMATIC OUPUT BUFFERING DESIRED

P: =1 IF DRIVER TO HANDEL POWER FAIL RECOVERY.

O: =1 IF DRIVER TO HANDEL TIME OUT.

T: DEVICE TIME-OUT BIT - CLEARED BEFORE EACH

IO INITIATION; SET IF DEVICE TIMES-OUT.

UNIT#: OPTIONAL FOR DEVICES REQUIRING

SUB-CHANNEL DESIGNATION

CHANNEL#: I/O SELECT CODE (LOWER # IF

MULTI-BOARD INTERFACE)

AV (AVAILABILITY INDICATOR):

(SEE ALSO DRT FOR LU'S)

=0, UNIT AVAILABLE FOR OPERATION

=1, UNIT DISABLED

=2, UNIT CURRENTLY IN OPERATION

=3, UNIT WAITING FOR DMA CHANNEL

TYPE CODE: CODE IDENTIFYING TYPE OF I/O DEVICE

UNIT STATUS: ACTUAL OR SIMULATED UNIT STATUS

AT END OF OPERATION

## SPECIFICATION

LOCATION BASE PAGE 1650

# OF ENTRIES BASE PAGE 1651

## EQT EXAMPLE

EQT 13	DVR 05			
WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	35400	0	0	
2	35401	25042	10786	*"
3	35402	25132	10842	*Z
4	35403	40026	16406	@H
5	35404	2400	1280	
6	35405	0	0	
7	35406	0	0	
8	35407	0	0	
9	35410	0	0	
10	35411	0	0	
11	35412	0	0	
12	35413	15	13	
13	35414	36127	15447	<W
14	35415	150437	-12001	
15	35416	0	0	

EXTENT				
WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	36127	0	0	
2	36130	0	0	
3	36131	0	0	
4	36132	0	0	
5	36133	0	0	
6	36134	0	0	
7	36135	0	0	
8	36136	0	0	
9	36137	0	0	
10	36140	0	0	
11	36141	0	0	
12	36142	0	0	
13	36143	0	0	

# DEVICE REFERENCE TABLE (DRT)

THE DEVICE REFERENCE TABLE PROVIDES LOGICAL ADDRESSING OF PHYSICAL UNITS DEFINED IN THE EQUIPMENT TABLE.

THE 'DRT' ENTRIES FOR THE LU'S CONSISTS OF 2- TABLES. THE LENGTH OF EACH TABLE IS DEFINED IN 'LUMAX'. TABLE 1 IS A 1- WORD ENTRY CORRESPONDING TO THE RANGE OF USER-SPECIFIED "LOGICAL" UNITS, 1 TO N WHERE N IS LT OR = TO 63(10). THE CONTENTS OF THE WORD CORRESPONDING TO A LOGICAL UNIT IS THE RELATIVE POSITION OF THE EQT ENTRY DEFINING THE ASSIGNED PHYSICAL UNIT:

TABLE 2 HAS A 1-WORD ENTRY FOR EACH LU DEFINED IN TABLE 1 AND CONTAINS THE STATUS OF THE CORRESPONDING LU. HIS POSITION IN MEMORY IS RIGHT AFTER TABLE 1. EACH ENTRY MAY REPRESENT 4 STATES.

	BIT 15	BITS 0-14	
STATE 1	0	0	=LU UP
STATE 2	1	0	=LU DOWN (NO STACKED I/O)
STATE 3	1	ADR	=LU DOWN STACKED I/O
STATE 4	1	LU	=LU DOWN I/O STACKED ON 2. LU

## TABLE 1 ENTRY:

15	11 10	6 5	0
!	SUB CHANNEL	! LOCKING RN #	! EQT NUMBER

## TABLE 2 ENTRY:

15	14	0
!UP/DN !	ADDRESS OR LU POINTER OR 0	!

## SPECIFICATION:

LOCATION	BASE PAGE 1652	DRT
LENGTH	BASE PAGE 1653	LUMAX

# INTERRUPT TABLE:

1 WORD PER SPECIFIED SELECT CODE. CONTENTS:

(+) = ADDRESS OF FIRST WORD OF EQT ENTRY

(-) = ADDRESS OF PROGRAM ID SEGMENT

0 = NO ENTRY

## SPECIFICATION:

LOCATION            BASE PAGE   1654   INTBA

#ENTRIES           BASE PAGE   1655   INTLG

FIRST ENTRY IS FOR S.C. = 6 (DMA 1)  
THE FIRST TWO WORDS ARE DYNAMIC (AS DMA ASSIGNMENT CHANGES)



# INTERRUPT TABLE EXAMPLE

## INT TABLE

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
6	36545	0	0	
7	36546	0	0	
10	36547	35323	15059	
11	36550	35342	15074	
12	36551	35361	15089	
13	36552	35246	15014	
14	36553	0	0	
15	36554	35152	14954	:J
16	36555	35152	14954	:J
17	36556	35265	15029	
20	36557	35265	15029	
21	36560	35133	14939	:I
22	36561	35114	14924	:L
23	36562	35171	14969	:Y
24	36563	35227	14999	
25	36564	35210	14984	
26	36565	140670	-15944	
27	36566	35417	15119	:I
30	36567	140670	-15944	
31	36570	140670	-15944	
32	36571	35645	15269	
33	36572	35664	15284	
34	36573	0	0	
35	36574	140670	-15944	
36	36575	140670	-15944	
37	36576	140670	-15944	
40	36577	140670	-15944	
41	36600	140670	-15944	
42	36601	140670	-15944	
43	36602	35626	15254	
44	36603	0	0	
45	36604	35304	15044	
46	36605	0	0	
47	36606	0	0	
50	36607	0	0	
51	36610	0	0	
52	36611	0	0	
53	36612	0	0	
54	36613	0	0	
55	36614	0	0	
56	36615	0	0	
57	36616	0	0	
60	36617	0	0	
61	36620	0	0	
62	36621	0	0	
63	36622	0	0	
64	36623	0	0	
65	36624	0	0	
66	36625	0	0	
67	36626	0	0	
70	36627	35703	15299	
71	36630	35722	15314	
72	36631	35741	15329	
73	36632	35760	15344	
74	36633	35777	15359	
75	36634	36016	15374	<N
76	36635	36035	15389	<I
77	36636	36054	15404	<.

TRACK ASSIGNMENT TABLE (TAT):

1 WORD PER TRACK ON LU 2 AND LU 3:

CONTENTS:

- ID SEGMENT ADDRESS OF PROGRAM-OWNER
- 077777 FOR GLOBAL ASSIGNMENT
- 077776 FOR FMP ASSIGNMENT
- 100000 FOR SYSTEM ASSIGNMENT
- 0 FOR AVAILABLE

SPECIFICATION:

LOCATION	BASE PAGE 1656	TAT
* ENTRIES	BASE PAGE 1755	TATLG
* TRACKS ON LU2	BASE PAGE 1756	TATSD
* SECTORS/TRACK, LU2	BASE PAGE 1757	SECT2
* SECTORS/TRACK, LU3	BASE PAGE 1760	SECT3

# CLASS TABLE:

-----

THE CLASS TABLE ENTRY CAN BE IN ONE OF FOUR DIFFERENT STATES:

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
! 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 !
-----

```

STATE 1: CLASS DEALLOCATED, AVAILABLE

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
! 0 ! ADDRESS OF FIRST ENTRY !
-----

```

STATE 2: POINTER TO FIRST ENTRY IN CLASS QUEUE

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
! 1 0 X! SECURITY CODE ! NUMBER OF PENDING REQS. !
-----

```

STATE 3: CLASS ALLOCATED, NO ONE WAITING ON CLASS  
NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
-----
! 1 1 X! SECURITY CODE ! NUMBER OF PENDING REQS. !
-----

```

STATE 4: CLASS ALLOCATED, SOMEONE WAITING (SUSPENDED)  
NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

## CLASS QUEUE FORMAT:

WORD	CONTENTS
----	-----
1	< LINKAGE WORD >
2	<T, CONTROL INFO, CODE >
3	<PRIORITY OF REQUESTOR > (CHANGED TO STATUS AT COMP.)
4	<TOTAL BLOCK LENGTH WORDS>
5	<CLASS ID WORD >
6	<USER BUFFER LENGTH > (CHANGED TO TLOG AT COMP.)
7	< OPTIONAL PARAMETER 1 >
8	< OPTIONAL PARAMETER 2 >
9	<WORD 1 OF USER BUFFER >
N+8	<WORD N OF USER BUFFER >

THE <T> FIELD (BITS 15-14 IN CONTROL WORD)

IDENTIFIES THE REQUEST TYPE AS:

```

00 USER (NORMAL OPERATION)
01 USER (AUTOMATIC BUFFERING)
10 SYSTEM
11 CLASS I/O

```

SPECIFICATION: \$CLAS = # ENTRIES IN CLASS TABLE  
HEADS CLASS TABLE

# L.U. SWITCH TABLE:

THE L.U. SWITCH TABLE HAS ONE WORD PER ENTRY:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEW L.U. - 1								ORIGINAL LU - 1							

## SPECIFICATION:

\$LUSW HEADS THE TABLE  
= # ENTRIES IN THE TABLE

# RESOURCE NUMBER TABLE:

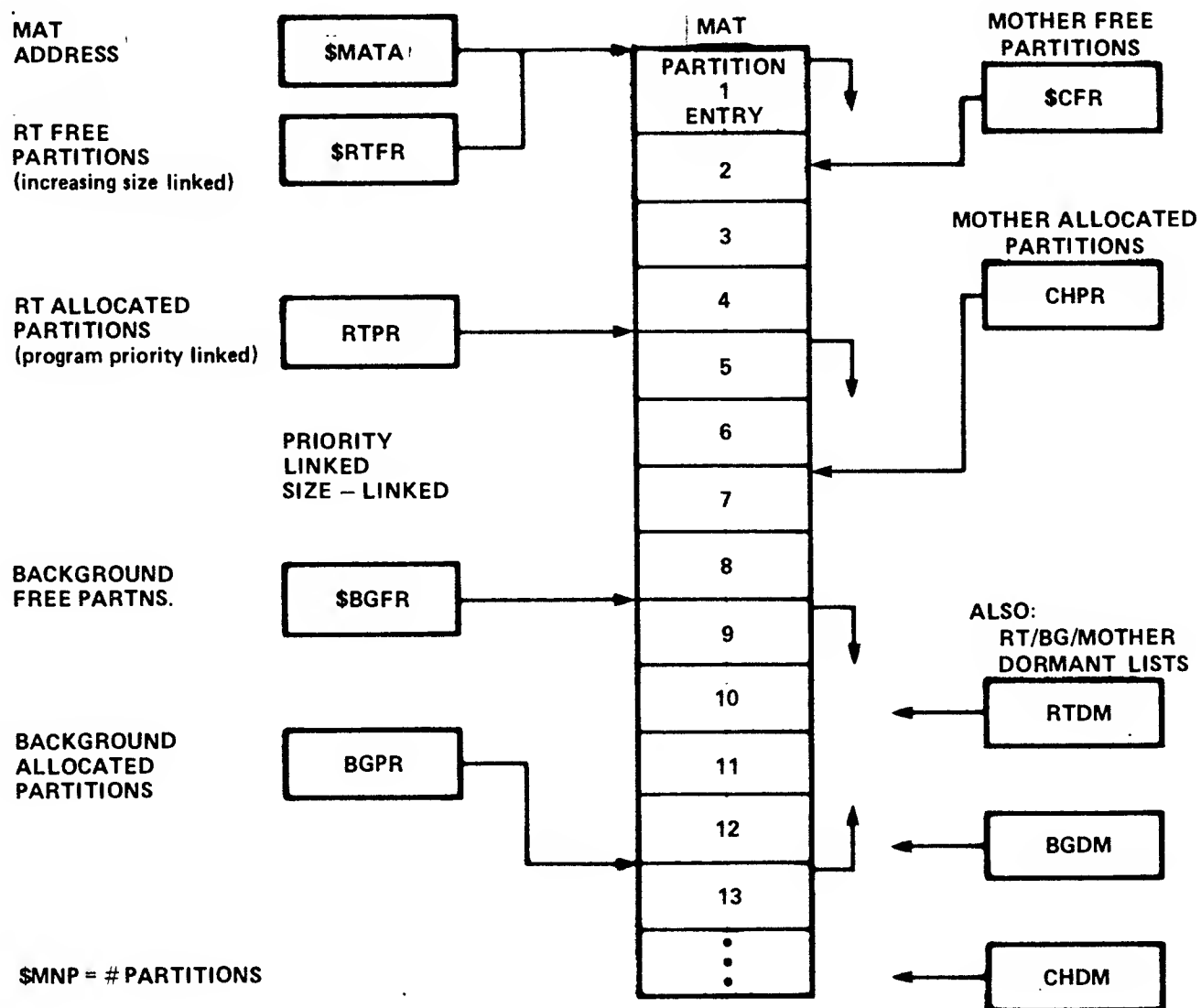
RESOURCE NUMBER TABLE HAS ONE WORD PER ENTRY:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWNER'S ID NUMBER								LOCKER'S ID NUMBER							
OR								OR							
377B FOR GLOBAL ALLOCATE								377B FOR GLOBAL LOCK							
OR								OR							
0 FOR AVAILABLE								0 FOR UNLOCKED							

## SPECIFICATION:

\$RNTB HEADS THE TABLE AND  
= NUMBER OF ENTRIES

# MEMORY ALLOCATION TABLE (MAT)



(NOTE: listheads are kept in Table Area 2 and DISPM)

- last entry points to start of allocated list
- priority linked

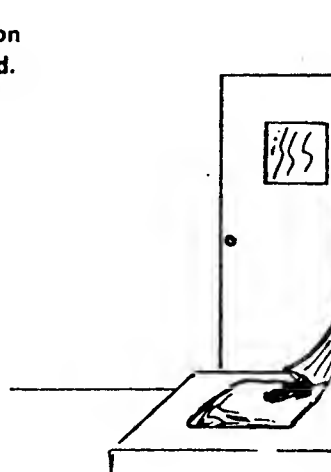
# MAT ENTRY

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Linkage pointer to next entry (-1 if undefined partition)															
1	Priority of current resident program															
2	Current resident's ID-segment address															
3	M		D													
4	R	C														
5	RT														S	
6	Subpartition Link Word (SLW)															

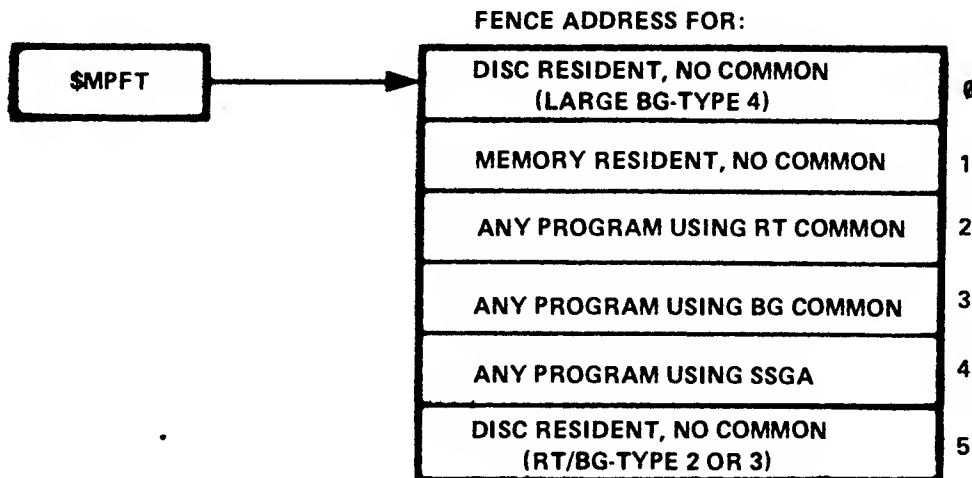
- D = Resident is dormant — save resources, serially reusable, or operator suspended
  - M = Mother partition
  - R = Partition is reserved
  - C = Partition is part of a chained mother partition
  - RT = REAL TIME PARTITION
  - S = Program's dispatching status
    - 0 - Read in progress
    - 1 - Program is resident
    - 2 - SWAP out or segment load in progress
    - 3 - SWAP out complete but program still resident
    - 4 - Subpartition swap-out started for mother partition
    - 5 - Subpartition swap-out completed. Mother cleared.
- "NORMAL" sequence: 2,3,0,1

## SLW

- = 0 - Partition not a subpartition
- = Next subpartition
- = Mother partition if partition is last subpartition



## MEMORY PROTECT FENCE TABLE (MPFT)



- EACH PROGRAM HAS A MPFT INDEX (MPFI) IN ID SEG. WORD 22
- CURRENT MEMORY PROTECT FENCE IS IN BP WORD FENCE (1775).

# DRIVER MAPPING TABLE

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORD 1 OF DMT ENTRY FOR EQT ENTRY:	1	SD	(RESERVED)						M								
	2	SD	(RESERVED)						M								
	3	SD	(RESERVED)						M								
								.									
	N	SD	(RESERVED)						M								
WORD 2 OF DMT ENTRY FOR EQT ENTRY:	1	MR	(RESERVED)						P								
	2	MR	(RESERVED)						P								
								.									
								.									
	N	MR	(RESERVED)						P								

WHERE:

SD = 0 IMPLIES DRIVER RESIDES IN A DRIVER PARTITION, AND  
M = STARTING PAGE NUMBER OF PARTITION IN BITS 0-9

SD = 1 IMPLIES DRIVER RESIDES IN SYSTEM DRIVER AREA, AND  
M = 0 IMPLIES DRIVER NOT DOING ITS OWN MAPPING  
M = 1 IMPLIES DRIVER DOING ITS OWN MAPPING

MR = 1 IMPLIES THAT THE I/O REQUEST BUFFER IS LOCATED IN  
A MEMORY RESIDENT PROGRAM.  
(P VALUE NOT SIGNIFICANT – RESERVED FOR FUTURE USE)

MR = 0 IMPLIES THAT THE I/O REQUEST BUFFER IS NOT LOCATED  
IN A MEMORY RESIDENT PROGRAM. BUFFER LOCATION IS  
INDICATED BY THE VALUE OF P, AS FOLLOWS:  
P = 0 IMPLIES BUFFER IS IN THE SYSTEM AREA  
P NOT ZERO IMPLIES BUFFER IS LOCATED IN A DISC  
RESIDENT PROGRAM. P IS THE PHYSICAL  
PAGE NUMBER OF THE PROGRAM'S BASE PAGE

N = NUMBER OF EQT ENTRIES IN SYSTEM (BP 1651)

\$DVMP = address of driver mapping table

\$DVPT = logical start page of driver partition

\$DLTH = # pages per driver partition



# DRIVER MAPPING TABLE EXAMPLE

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	2714	2	2	
2	2715	30	24	
3	2716	32	26	
4	2717	36	30	
5	2720	30	24	
6	2721	2	2	
7	2722	30	24	
8	2723	2	2	
9	2724	34	28	
10	2725	34	28	
11	2726	34	28	
12	2727	2	2	
13	2730	34	28	
14	2731	30	24	
15	2732	100001	-32767	
16	2733	100001	-32767	
17	2734	100001	-32767	
18	2735	100001	-32767	
19	2736	100001	-32767	
20	2737	64	52	4
21	2740	0	0	
22	2741	0	0	
23	2742	0	0	
24	2743	0	0	
25	2744	0	0	
26	2745	0	0	
27	2746	0	0	
28	2747	0	0	
29	2750	0	0	
30	2751	0	0	
31	2752	0	0	
32	2753	0	0	
33	2754	0	0	
34	2755	0	0	
35	2756	0	0	
36	2757	0	0	
37	2760	0	0	
38	2761	0	0	

# TRACK MAP TABLES

		WORD
7900 DISC	\$TB31	NEG. # OF 64 WRD SECT./TRK
		0
		FIRST TRACK, SUBCHANNEL 0
		1
		FIRST TRACK, SUBCHANNEL 1
		2
		FIRST TRACK, SUBCHANNEL 2
		3
		FIRST TRACK, SUBCHANNEL 3
		4
		FIRST TRACK, SUBCHANNEL 4
		5
		FIRST TRACK, SUBCHANNEL 5
		6
		FIRST TRACK, SUBCHANNEL 6
		7
		FIRST TRACK, SUBCHANNEL 7
		8
		# OF TRACKS, SUBCHANNEL 0
		9
		# OF TRACKS, SUBCHANNEL 1
		10
		# OF TRACKS, SUBCHANNEL 2
		11
		# OF TRACKS, SUBCHANNEL 3
		12
		# OF TRACKS, SUBCHANNEL 4
		13
		# OF TRACKS, SUBCHANNEL 5
		14
		# OF TRACKS, SUBCHANNEL 6
		15
		# OF TRACKS, SUBCHANNEL 7
		16

## 7905/7906/7920 DISC

		WORD
	\$TB32	NEG. # OF TOTAL SUBCHANNELS
		1
		STARTING CYLINDER #
		2
SUBCHANNEL 0		# SURFACES, HEAD, UNIT
		3
		# TRACKS
		4
:		:
		:
		:
		:
SUBCHANNEL n		STARTING CYLINDER #
		3n-1
		# SURFACES, HEAD, UNIT
		3n
		# TRACKS
		3n+1

# \$TB32 EXAMPLE

WORD	LOCATION	VALUE(R)	VALUE(10)	VALUE(AS)
1	2000	177772	- 6	
2	2001	0	0	
3	2002	11000	4608	
4	2003	400	256	
5	2004	403	259	
6	2005	11000	4608	
7	2006	226	150	
8	2007	0	0	
9	2010	20000	8192	
10	2011	313	203	
11	2012	147	103	
12	2013	20000	8192	
13	2014	313	203	
14	2015	316	206	
15	2016	20000	8192	
16	2017	313	203	
17	2020	465	309	5
18	2021	20000	8192	
19	2022	313	203	



## APPENDIX H



## SE LEVEL II HOMEWORK

<u>DAY (DUE)</u>	<u>ASSIGNMENT</u>
Mon (Wed am)	<p>1. Included in your material is a "User Program State Diagram". Indicate on the diagram one specific reason for a user program to make <u>each</u> legal state change. Types of reasons <u>include</u>:</p> <ul style="list-style-type: none"><li>Operator Commands</li><li>Exec Calls</li><li>Environment Events</li></ul>
Mon (Fri)	<p>2. Included in your material is a generation map listing. Indicate on the listing where each generator response is located in the final RTE system (i.e. which table, list, BP location, or global in operating system).</p>
Tues (Thur)	<p>3. Included in the course material is a RTE memory dump and two blank WHZAT printouts. Using the dump, fill in both WHZAT printouts. The partition list can be completed after Tuesday's lecture and the program state list after Wednesday (see Section 14 of the workbook for a sample WHZAT listing).</p>
Wed (Fri)	<p>4. Trace a "SS, PROGA" request from interrupt thru completion. See the "ON,XYZ" trace in Section 6 for an example. How does RTE implement an EXEC 7 request differently than a "SS" command?</p>

SE LEVEL II LABS  
(5 REQUIRED)

DAY

ASSIGNMENT

- Mon. (required)
1. Use CMM4 to find the following information about the training systems:
    - a. TBG select code and system console EQT entry address.
    - b. Starting address and number of EQT entries.
    - c. Location of and number of class numbers in the class table (\$CLAS).
    - d. Number of programs in the schedule and general wait lists.
    - e. Program name in ID segment #5.
    - f. List the MR map.
    - g. Memory address of \$CIC, \$LIST, and \$XEQ.
    - h. Value in trap cells 11 thru 15 (careful).
    - i. EQT and subchannel of LU 26.
    - j. ID segment address of currently executing program.
    - k. Size of BG and RT common.
    - l. Memory address of WHZAT and DVR32.
    - m. Number of tracks on LU2.
- Mon.
2.
    - a. Use CMM4 to determine the number of pages in each section of physical memory thru the SAM extension. (See Section 2 for a copy of physical memory.)
    - b. Program LABL2 sorts and calculates the average of 20 numbers. Compile it, load it and use only DBUGR to:
      - 1) Correct the cause of the DMS error.
      - 2) Correct the spelling of the second title to "SORTED ARRAY".
      - 3) Set a breakpoint in the sorting loop in the subroutine SORT. Use variations of the "n/P", "T", and "n/T" commands in the load and examine the array "NUMB" as it is sorted.
      - 4) Initialize "ISUM" to 0 instead of 999 by setting a breakpoint before it is initialized and modifying the A or B register.



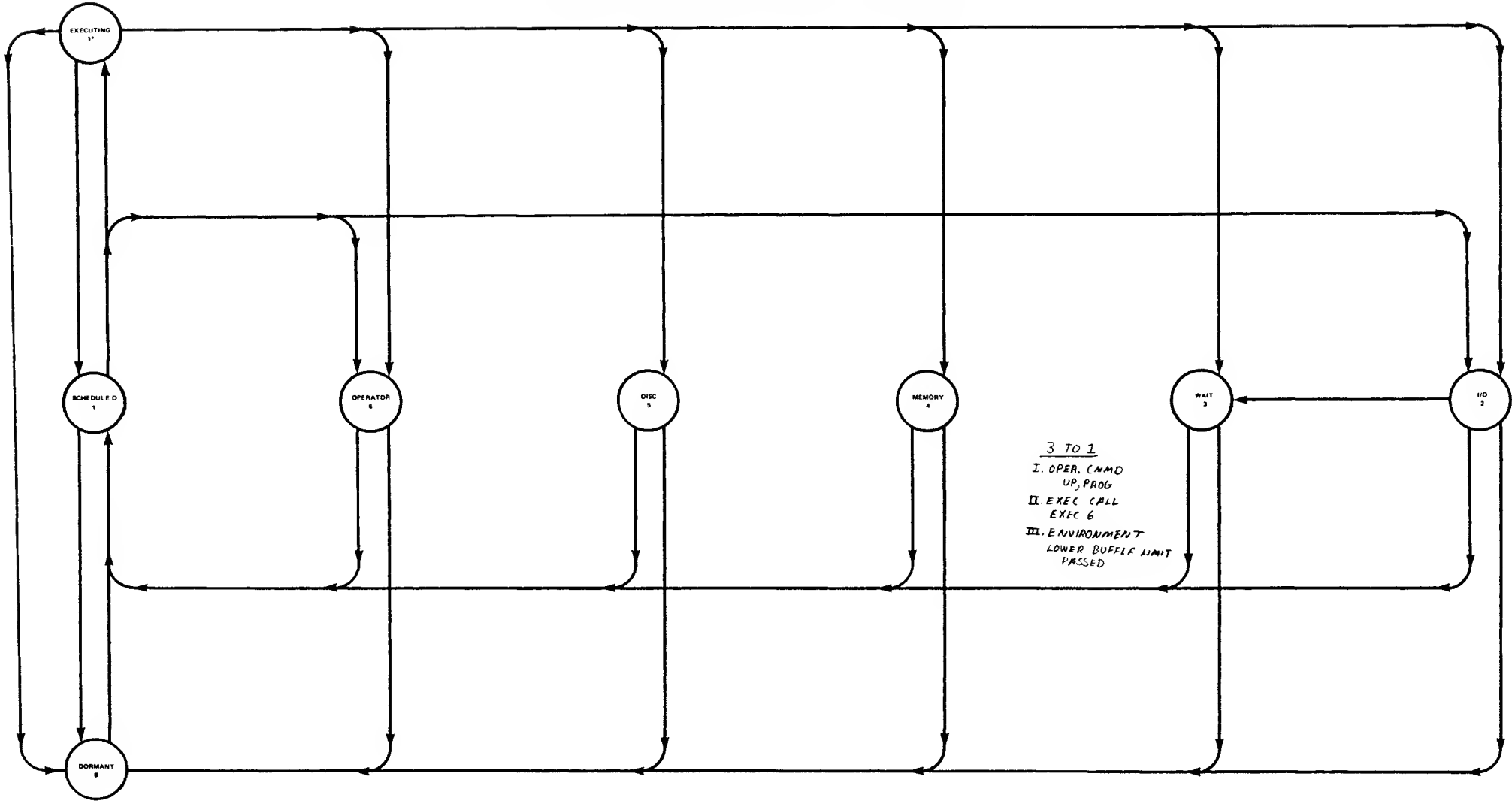
Day Material  
Presented

ASSIGNMENT

- |       |      |   |
|-------|------|---|
| Mon.  | 3.   | a. Without a generation map, find all the RP microcode values generated into the training systems. (Use the utilities.)<br>b. Find which loader ROM's (and their locations) are installed in the training systems. Use the front panel and "Loader ROM's" installation manual only. |
| Mon.  | 4.   | Write a program to list all four DMS maps and the DMS status and violation registers. You will need to make your routine privileged. See page 18-16 of work book.   |
| Tues. | 5.   | Write a program to search the MAT table and report the number of memory pages in the partition area.  |
| Tues. | * 6. | Write a program which will accept any EQT# and return the number of I/O requests queued on the EQT, the type of each request (user, buffered, class, or system), and the size of each request.  |
| Wed.  | * 7. | Patch \$CIC to keep a trace of the last 25 interrupt select codes (Hint" use a circular buffer to record the SC's.)   |
| Wed.  | 8.   | Patch PRMPT (on the disc) to change its prompt character from ">" to "?". Re-boot the system and verify the change. Restore PRMPT and boot the system when you finish.  |
| Thur. | 9.   | Write a program to list the number and size of each block in the free SAM list.   |
| Thur. | 10.  | Run program RENT. At each pause record the structure of the re-entrant list and TDB's linked to it.   |
| Thur. | 11.  | Write a reentrant subroutine to return total, average, min., max., and medium of a 10 element array.  |

\*Labs marked with an \* are equivalent to 2 labs.

USER PROGRAM STATE DIAGRAM



DEMOLF T=00003 IS ON CR00002 USING 00153 BLKS R=0000

```
0001 ECHO?
0002 YES * ECHO?
0003
0004 EST. # TRACKS IN OUTPUT FILE?
0005 *****
0006 *
0007 * DEMO SYSTEM GENERATION
0008 *
0009 * 3-21-78 *
0010 *
0011 *****
0012 40 * EST. # OF TRACKS IN OUTPUT FILE?
0013
0014 OUTPUT FILE NAME?
0015 DEMOSY:114, * OUTPUT FILE NAME?
0016
0017 SYSTEM DISC?
0018 7905 * SYSTEM DISC?
0019
0020 CONTROLLER SELECT CODE?
0021 12 * CONTROLLER SELECT CODE?
0022
0023 # TRKS, FIRST CYL #, HEAD #, # SURFACES, UNIT, # SPARES FOR SUBCHNL:
0024 00?
0025 * # TRKS, FIRST TRK ON SUBCHNL:
0026 256,0,2,1,0,3 * SUBCHANNEL 0
0027 01?
0028 150,259,2,1,0,2 * SUBCHANNEL 1
0029 02?
0030 203,0,0,2,0,3 * SUBCHANNEL 2
0031 03?
0032 203,103,0,2,0,3 * SUBCHANNEL 3
0033 04?
0034 203,206,0,2,0,3 * SUBCHANNEL 4
0035 05?
0036 203,309,0,2,0,1 * SUBCHANNEL 5
0037 06?
0038 /E
0039
0040 SYSTEM SUBCHNL?
0041 0 * SYSTEM SUBCHNL?
0042
0043 AUX DISC (YES OR NO OR # TRKS)?
0044 NO * AUX. DISC
0045
0046 TRG SELECT CODE?
0047 11 * TRG SELECT CODE?
0048
0049 PRIV. INT. SELECT CODE?
0050 0 * PRIV. INT. SELECT CODE?
0051
0052 MEM. RES. ACCESS TABLE AREA II?
0053 YES * MEM. RES. ACCESS TABLE AREA II?
0054
0055 RT MEMORY LOCK?
0056 YES * RT MEMORY LOCK?
0057
0058 BG MEMORY LOCK?
```

0059	YES	* BG MEMORY LOCK?
0060		
0061	SWAP DELAY?	
0062	50	* SWAP DELAY?
0063		
0064	MEM SIZE?	
0065	128	* MEM SIZE?
0066		
0067	BOOT FILE NAME?	
0068	0	* BOOT FILE NAME?
0069		
0070		
0071	PROG INPUT PHASE:	
0072	-	
0073	*****	
0074	-	
0075	*** PROGRAM INPUT PHASE ***	
0076	-	
0077	*****	
0078	-	
0079	MAP MODULES	
0080	-	
0081	LINKS IN CURRENT	
0082	-	
0083	*****	
0084	-	
0085	** STANDARD RTE-IV MODULES **	
0086	-	
0087	*****	
0088	-	
0089	REL,XCR4SY::1904,	* RTE-IV OPERATING SYSTEM
0090	-	
0091	*****	
0092	-	
0093	** DRIVERS **	
0094	-	
0095	*****	
0096	-	
0097	REL,XDVR32::1904,	* 7905 DISC
0098	-	
0099	REL,XDVR12::1904,	* LINE PRINTER(2767)
0100	-	
0101	REL,X4DV05::1904,	* 2645 CRT
0102	-	
0103	REL,XDVR36::1904,	* WCS DRIVER
0104	-	
0105	REL,XDVR23::1904,	* 7970 MAG. TAPE
0106	-	
0107	REL,XDVA13::1904,	* TV MONITOR
0108	-	
0109	REL,XDVR07::1904,	* MULTIPoint
0110	-	
0111	REL,X2DV37::1904,	* HPIB
0112	-	
0113	REL,X4DP43::1904,	* POWER FAIL
0114	-	
0115	REL,XDVA12::1904,	* LP (2607)
0116	-	
0117	*****	
0118	-	

0119	★★ SPECIAL SYSTEM MODULES ★★	
0120	-	
0121	*****	
0122	-	
0123	REL,XAUTO7::1904,	★ DVR07 POWER FAIL ROUTINE
0124	-	
0125	*****	
0126	-	
0127	★★ USER PROGRAMS ★★	
0128	-	
0129	*****	
0130	-	
0131	REL,XHPIB::1904,	★ HPID DEVICE SUBROUTINE
0132	-	
0133	REL,X4LDR::1904,	★ RELOCATING LOADER
0134	-	
0135	REL,XGASP4::1904,	★ GASP
0136	-	
0137	REL,X4SPOL::1904,	★ SPOOLING
0138	-	
0139	REL,XBMPG::1904,	★ FMGR MODIFIED FOR RTE-IV
0140	-	
0141	REL,X4MTM::1904,	★ MULTI-TERMINAL MONITOR
0142	-	
0143	*****	
0144	-	
0145	★★ LIBRARIES ★★	
0146	-	
0147	*****	
0148	-	
0149	REL,X4SYLB::1904,	★ RTE-IV SYSTEM LIBRARY
0150	-	
0151	REL,XBMLIB::1904,	★ RTE FILE MANAGEMENT LIBRARY
0152	-	
0153	REL,XCLIB ::1904,	★ RTE COMPILER LIBRARY
0154	-	
0155	REL,XFF4.N::1904,	★
0156	-	
0157	REL,XRLIB1::1904,	★
0158	-	
0159	REL,XRLIB2::1904,	★
0160	-	
0161	REL,XIMAG1::1904,	★ IMAGE LIBRARY
0162	-	
0163	REL,XIMAG2::1904,	
0164	-	
0165	REL,XDBGUR::1904,	★ USER DEBUG
0166	-	
0167	REL,XMPLIB::1904,	★ MULTIPOINT LIBRARY
0168	-	
0169	REL,XBASLB::1904,	★ BASIC
0170	-	
0171	REL,XMLIB::1904,,	★ BASIC NEW MESSAGE FILE
0172	-	
0173	REL,XTRAP::1904,	★ SRQ.T IN SSGA FOR BASIC
0174	-	
0175	*****	
0176	-	
0177	★★ UTILITIES ★★	
0178	-	

0179	*****	
0180	-	
0181	REL,X4WHZT::1904,	* WHZAT PROGRAM
0182	-	
0183	REL,XCMM4::1904,	* CMM4 PROGRAM
0184	-	
0185	REL,XLUMAP,	* INIT. LUI'S FOR DEMO
0186	-	
0187	REL,XKYDMP::1904,	* KEY DUMP UTIL.
0188	-	
0189	REL,XUTIL::1904,	* SET UP SSGA FOR TVST+TVMEM
0190	-	
0191	REL,XDSPMP::1904,	* MP UTIL.
0192	-	
0193	REL,XEXMP::1904,	* MP UTIL.
0194	-	
0195	REL,XLGAT::1904,	* TRACK ASSIGNMENT LISTING
0196	-	
0197	DISPLAY UNDEFS	
0198	UNDEFS	
0199	&6940	
0200	-	
0201	/E	
0202	UNDEFS	
0203	&6940	
0204		
0205	PARAMETERS	
0206	-	
0207	*****	
0208	*** PARAMETER INPUT PHASE ***	
0209	*****	
0210	D.RTR,1,1	
0211	-	
0212	CMM4,1,90	
0213	-	
0214	WHZAT,1,1	
0215	-	
0216	AUTOR,19	
0217	-	
0218	PRMPT,1,10	
0219	-	
0220	RSPN1,1,10	
0221	-	
0222	LOADR,3,80	
0223	-	
0224	EXTND,17	
0225	-	
0226	SMP,18	
0227	-	
0228	SPOUT,18	
0229	-	
0230	JOB,18	
0231	-	
0232	GASP,19	
0233	-	
0234	.DBRN,30	* SSGA FOR IMAGE
0235	-	
0236	TRAP,30	* SSGA FOR BASIC
0237	-	
0238	TTYEV,17	* FOR SSGA ACCESS

```

0239 -
0240 IDGET,17
0241 -
0242 /E
0243
0244 CHANGE ENTS?
0245 -
0246 *****
0247 ** CHANGE ENTS? **
0248 *****
0249 .MPY,RP,100200
0250 -
0251 .DIV,RP,100400
0252 -
0253 .DLD,RP,104200
0254 -
0255 .DST,RP,104400
0256 -
0257 .FAD,RP,105000
0258 -
0259 .FSB,RP,105020
0260 -
0261 .FMP,RP,105040
0262 -
0263 .FDV,RP,105060
0264 -
0265 IFIX,RP,105100
0266 -
0267 FLOAT,RP,105120
0268 -
0269 DBLE,RP,105201
0270 -
0271 SNGL,RP,105202
0272 -
0273 .XMPY,RP,105203
0274 -
0275 .XDIV,RP,105204
0276 -
0277 .DFEK,RP,105205
0278 -
0279 .XADD,RP,105213
0280 -
0281 .XSUB,RP,105214
0282 -
0283 .GOTO,RP,105221
0284 -
0285 .,MAP,RP,105222
0286 -
0287 .ENTR,RP,105223
0288 -
0289 .ENTP,RP,105224
0290 -
0291 .PWR2,RP,105225
0292 -
0293 .FLUN,RP,105226
0294 -
0295 .SETP,RP,105227
0296 -
0297 .PACK,RP,105230
0298 -

```

★ "

★ EAU MACRO'S

★ HFP MACRO'S

★ FFP MACRO'S

```

0299 .XFER,RP,105220
0300 -
0301 .XPAK,RP,105206
0302 -
0303 XADD,RP,105207
0304 -
0305 XSUB,RP,105210
0306 -
0307 XMPY,RP,105211
0308 -
0309 XDIV,RP,105212
0310 -
0311 .XCOM,RP,105215
0312 -
0313 .,DCM,RP,105216
0314 -
0315 DDINT,RP,105217
0316 -
0317 MVW.,RP,105777 * 21MX EXTENDED INSTRUCTION SET
0318 -
0319 .MVW,RP,105777
0320 -
0321 .EMAP,RP,105257 * 21XE EMA
0322 -
0323 .EMIO,RP,105240
0324 -
0325 MMAP,RP,105241
0326 -
0327 /E
0328
0329
0330
0331 TABLE AREA I
0332
0333
0334 EQUIPMENT TABLE ENTRY
0335
0336 EQT 01?
0337 *****
0338 *** TABLE GENERATION PHASE***
0339 *****
0340 * TABLE AREA I *
0341 *****
0342 * EQUIPMENT TABLE ENTRY *
0343 *****
0344 12,DVR32,D * EQT 01 = 7905 DISC
0345
0346 EQT 02?
0347 21,DVR05,T=32767,X=13 * EQT 02 = SYSTEM CONSOLE (2648)
0348
0349 EQT 03?
0350 10,DVR36,D * EQT 03 = WCS
0351
0352 EQT 04?
0353 20,DVR37,T=6000,X=25 * EQT 04 = HPTB
0354
0355 EQT 05?
0356 23,DVR05,T=32767,X=13 * EQT 05 = AUX. 2645 OR 2648
0357
0358 EQT 06?

```



0359 24,DVR12,B,T=32767  
 0360  
 0361 EQT 07?  
 0362 25,DVA12,B,T=32767  
 0363  
 0364 EQT 08?  
 0365 16,DVR23,D,T=32767  
 0366  
 0367 EQT 09?  
 0368 22,DVR07,X=5  
 0369  
 0370 EQT 10?  
 0371 77,DVR07,X=5  
 0372  
 0373 EQT 11?  
 0374 77,DVR07,X=5  
 0375  
 0376 EQT 12?  
 0377 15,DVA13,D,T=20  
 0378  
 0379 EQT 13?  
 0380 77,DVR07,X=5  
 0381  
 0382 EQT 14?  
 0383 26,DVR05,B,T=32767,X=13  
 0384  
 0385 EQT 15?  
 0386 60,DVS43,S,M,X=18  
 0387  
 0388 EQT 16?  
 0389 61,DVS43,S,M,X=18  
 0390  
 0391 EQT 17?  
 0392 62,DVS43,S,M,X=18  
 0393  
 0394 EQT 18?  
 0395 63,DVS43,S,M,X=18  
 0396  
 0397 EQT 19?  
 0398 4,DVP43,M  
 0399  
 0400 EQT 20?  
 0401 /E  
 0402  
 0403  
 0404  
 0405  
 0406  
 0407  
 0408  
 0409  
 0410  
 0411  
 0412  
 0413  
 0414  
 0415  
 0416  
 0417  
 0418

# DEVICE REFERENCE TABLE

1 = EQT #?  
 \*\*\*\*\*  
 \* DEVICE REFERENCE TABLE \*  
 \*\*\*\*\*  
 2,0  
  
 2 = EQT #?  
 1,0  
  
 3 = EQT #?  
 0  
  
 4 = EQT #?

\* EQT 06 = LINE PRINTER(2767)  
  
 \* EQT 07 = LP(2607)  
  
 \* EQT 08 = MAG. TAPE  
  
 \* EQT 09 = MP LINE CONTROL  
  
 \* EQT 10 = MP TERMINAL #1  
  
 \* EQT 11 = MP TERMINAL #2  
  
 \* EQT 12 = TV  
  
 \* EQT 13 = MP TERMINAL #3  
  
 \* EQT 14 = AUX. TERMINAL (2645 OR 2648)  
  
 \* EQT 15 = SPOOL  
  
 \* EQT 16 = SPOOL  
  
 \* EQT 17 = SPOOL  
  
 \* EQT 18 = SPOOL  
  
 \* EQT 19 = POWER FAIL

\* LU 1 = SYSTEM CONSOLE (2648)  
  
 \* LU 2 = SYSTEM DISC (SUB. 0)  
  
 \* LU 3 = BIT BUCKET

0419 2,1  
 0420  
 0421 5 = EQT #?  
 0422 2,2  
 0423  
 0424 6 = EQT #?  
 0425 2,0  
 0426  
 0427 7 = EQT #?  
 0428 2,3  
 0429  
 0430 8 = EQT #?  
 0431 8,0  
 0432  
 0433 9 = EQT #?  
 0434 9,0  
 0435  
 0436 10 = EQT #?  
 0437 10,0  
 0438  
 0439 11 = EQT #?  
 0440 11,0  
 0441  
 0442 12 = EQT #?  
 0443 12,1  
 0444  
 0445 13 = EQT #?  
 0446 13,0  
 0447  
 0448 14 = EQT #?  
 0449 1,1  
 0450  
 0451 15 = EQT #?  
 0452 0  
 0453  
 0454 16 = EQT #?  
 0455 5  
 0456  
 0457 17 = EQT #?  
 0458 5  
 0459  
 0460 18 = EQT #?  
 0461 5  
 0462  
 0463 19 = EQT #?  
 0464 5  
 0465  
 0466 20 = EQT #?  
 0467 0  
 0468  
 0469 21 = EQT #?  
 0470 1,2  
 0471  
 0472 22 = EQT #?  
 0473 4,0  
 0474  
 0475 23 = EQT #?  
 0476 4,1  
 0477  
 0478 24 = EQT #?

\* LU 4 = SYS CONSOLE LEFT CTU  
  
 \* LU 5 = RT. CTU  
  
 \* LU 6 = SYS CONSOLE(2648)  
  
 \* LU 7 = 2648 GRAPHICS  
  
 \* LU 8 = MAG. TAPE  
  
 \* LU 9 = MP LINE CONTROL  
  
 \* LU 10 = MP TERM. #1  
  
 \* LU 11 = MP TERM. #2  
  
 \* LU 12 = TV MONITOR  
  
 \* LU 13 = MP TERM. #3  
  
 \* LU 14 = AUX. DISC (SUB. 1)  
  
 \* LU 15 = HIT BUCKET  
  
 \* LU 16 = AUX. TERM.  
  
 \* LU 17 = LT. CTU  
  
 \* LU 18 = RT. CTU  
  
 \* LU 19 = GRAPHICS  
  
 \* LU 20 = HIT BUCKET  
  
 \* LU 21 = AUX. DISC  
  
 \* LU 22 = MP IB  
  
 \* LU 23 = MP IB DEVICE #1

0479	4,2
0480	
0481	25 = EQT #?
0482	4,3
0483	
0484	26 = EQT #?
0485	1,3
0486	
0487	27 = EQT #?
0488	14,0
0489	
0490	28 = EQT #?
0491	14,1
0492	
0493	29 = EQT #?
0494	14,2
0495	
0496	30 = EQT #?
0497	14,3
0498	
0499	31 = EQT #?
0500	1,4
0501	
0502	32 = EQT #?
0503	3,0
0504	
0505	33 = EQT #?
0506	3,1
0507	
0508	34 = EQT #?
0509	15,0
0510	
0511	35 = EQT #?
0512	16,0
0513	
0514	36 = EQT #?
0515	17,0
0516	
0517	37 = EQT #?
0518	18,0
0519	
0520	38 = EQT #?
0521	6,0
0522	
0523	39 = EQT #?
0524	7,0
0525	
0526	40 = EQT #?
0527	1,5
0528	
0529	41 = EQT #?
0530	19
0531	
0532	42 = EQT #?
0533	/E
0534	
0535	
0536	INTERRUPT TABLE
0537	
0538	-

* LU 24 = HP IB DEVICE #2
* LU 25 = HP IB DEVICE #3
* LU 26 = AUX. DISC
* LU 27 = EXTRA 2648
* LU 28 = LT. CTU
* LU 29 = RT. CTU
* LU 30 = CRAPHS
* LU 31 = AUX DISC
* LU 32 = 1K WCS, LOWER
* LU 33 = 1K WCS, UPPER
* LU 34 = SPOOL
* LU 35 = SPOOL
* LU 36 = SPOOL
* LU 37 = SPOOL
* LU 38 = LP(2767)
* LU 39 = LP(2607)
* LU 40 = AUX DISC
* LU 41 = POWER FAIL

```

0539 *****
0540 * INTERRUPT TABLE *
0541 *****
0542 4,ENT,$POWR * POWER FAIL/AUTO RESTART
0543 -
0544 12,EQT,1 * 7905 DISC
0545 -
0546 13,EQT,12 * TV
0547 -
0548 14,EQT,12 * TV
0549 -
0550 15,EQT,12 * TV
0551 -
0552 16,EQT,8 * MAG. TAPE
0553 -
0554 17,EQT,8 * MAG. TAPE
0555 -
0556 20,EQT,4 * HP IB
0557 -
0558 21,PRG,PRMPT * SYSTEM CONSOLE (2648)
0559 -
0560 22,PRG,PRMPT * MP LINE CONTROL
0561 -
0562 23,PRG,PRMPT * AUX. TERM. 2648
0563 -
0564 24,EQT,6 * LP 2767
0565 -
0566 25,EQT,7 * LP 2607
0567 -
0568 26,PRG,PRMPT * AUX. TERMINAL 2648
0569 -
0570 77,ABS,0 * MULTIPOINT TERMINALS
0571 -
0572 /E
0573
0574
0575 TABLE AREA I MODULES
0576
0577 35TB1(0009)03176 03320 92067-16014 REV.1805 780223
0578
0579
0580 DRVR PART 00002
0581 CHANGE DRVR PART?
0582 *****
0583 *** SYSTEM BOUNDARIES PHASE ***
0584 *****
0585 0 * CHANGE DRIVER PARTITION?
0586
0587
0588 DP 011
0589
0590 DVR32(0009)04000 05525 92060-16031 REV 1805 780126
0591
0592 DVR12(0009)05553 06311 29028-60002 780103 REV 1805
0593
0594 DVR23(0009)06322 07165 92202-16001 REV. A
0595
0596 DVA13(0009)07166 07421 91200-16001 REV 1648 -- 761124
0597
0598

```

0599			
0600	SUBSYSTEM GLOBAL AREA		
0601			
0602	SP.CL	)10000 10002	92067-16028 REV.1805 780317
0603			
0604	.DBRN	)10003 10033	92063-12001 REV. 1805 770601
0605			
0606	TRAP	)10034 11072	92101-16010 770208
0607			
0608	UTIL	)11073 11106	
0609			
0610			
0611			
0612	RT COMMON	00000	
0613	CHANGE RT COMMON ?		
0614	100		* CHANGE RT COMMON?
0615	RT COM ADD	11107	
0616			
0617			
0618	BG COMMON	00341	
0619	CHANGE BG COMMON ?		
0620	0		* CHANGE BG COMMON?
0621	BG COM ADD	11253	
0622	BG COMMON	00341	
0623			
0624			
0625	SYSTEM DRIVER AREA		
0626			
0627	DVP43(0099)12000	12634	92067-16004 REV.1805 771219
0628			
0629	DVS43(0099)12653	15600	92067-16028 REV.1805 771110
0630			
0631			
0632			
0633	TABLE AREA II		
0634			
0635	# OF I/O CLASSES?		
0636	32		* # OF I/O CLASSES?
0637			
0638	# OF LU MAPPINGS?		
0639	24		* # OF LU MAPPINGS?
0640			
0641	# OF RESOURCE NUMBERS?		
0642	32		* # OF RESOURCE NUMBERS?
0643			
0644	BUFFER LIMITS (LOW, HIGH)?		
0645	100,400		* BUFFER LIMITS (LOW,HIGH)?
0646			
0647	# OF BLANK ID SEGMENTS?		
0648	50		* # OF BLANK ID SEGMENTS?
0649			
0650	# OF BLANK SHORT ID SEGMENTS?		
0651	35		* # OF BLANK SHORT ID SEGMENTS?
0652			
0653	# OF BLANK ID EXTENSIONS?		
0654	15		* # OF BLANK ID EXTENSIONS?
0655			
0656	MAXIMUM # OF PARTITIONS?		
0657	15		* MAXIMUM # OF PARTITIONS?
0658			

0659  
 0660 TABLE AREA II MODULES  
 0661  
 0662 SSTB2(0099)24317 24363 92067-16014 REV.1805 771107  
 0663  
 0664  
 0665  
 0666 SYSTEM  
 0667  
 0668 SCSY4(0099)24364 24363 92067-16014 REV.1805 780125  
 0669  
 0670 DISP4(0099)24444 32026 92067-16014 REV.1805 780317  
 0671  
 0672 RTIME(0099)32035 32641 92067-16014 REV.1805 780104  
 0673  
 0674 SAS4(0099)32642 32734 92067-16014 REV.1805 780125  
 0675  
 0676 RTIO4(0099)33016 40336 92067-16014 REV.1805 780310  
 0677  
 0678 EXEC4(0099)40356 42734 92067-16014 REV.1805 780310  
 0679  
 0680 STRN4(0099)42760 43132 92067-16014 REV.1805 780104  
 0681  
 0682 SCHD4(0099)43150 50327 92067-16014 REV.1805 780317  
 0683  
 0684 SANC (0099)50337 50544 92067-16014 REV.1805 741120  
 0685  
 0686 UCMD4(0099)50545 51706 92067-16014 REV.1805 771102  
 0687  
 0688 PERR4(0099)51716 52656 92067-16014 REV.1805 780227  
 0689  
 0690 SBMON(0099)52657 52656 92002-12001 REV.1805 771116  
 0691  
 0692 SYSLB(0099)52657 52656 92067-16035 REV.1805 770714  
 0693  
 0694 SBALB(0099)52657 52656 92002-16006 REV.1805 771116  
 0695  
 0696 FF4.A(0099)52657 52656 24998-16002 REV.1805 780303  
 0697  
 0698 RL1B1(0099)52657 52656 24998-16001 REV.1805 771116  
 0699  
 0700 RL1B2(0099)52657 52656 24998-16001 REV.1805 771116  
 0701  
 0702 MPL1B(0099)52657 52656 91730-12001 REV.1805 780301  
 0703  
 0704 SCNFG(0099)52710 57546 92067-16014 REV.1805 770112  
 0705  
 0706  
 0707 PARTITION DRIVERS  
 0708  
 0709 DP 02:  
 0710  
 0711 DVR05(0099)04056 06670 92001-16027 REV.1806 1-17-78  
 0712  
 0713 DVA12(0099)06763 07663 92001-16020 1806 780112  
 0714  
 0715 DP 03:  
 0716  
 0717 DVR36(0099)04014 06075 RTE DVR36 13197-16001 REV.A 751221  
 0718

```

0719 DP 04:
0720
0721 DVR07(0099)04070 06431 91730-16001 REV 1805 780307 &DV7D2
0722
0723 DP 05:
0724
0725 DVR37(0099)04065 06411 59310-16003 REV. 1805, 780306
0726
0727
0728 MEMORY RESIDENT LIBRARY
0729
0730 PRTN 26000 26112 92067-16035 REV.1805 771005
0731 TMVAL 26113 26202 92067-16035 REV.1805 770715
0732 IFBRK 26203 26232 92067-16035 REV.1805 770621
0733 PARSE 26233 26252 92067-16035 REV.1805 770714
0734 SPARS 26253 26473 92067-16035 REV.1805 770621
0735 CNUMD 26474 26513 92001-16035 REV.1805 770621
0736 CNUMO 26514 26533 92067-16035 REV.1805 770621
0737 SCVT3 26534 26621 92067-16035 REV.1805 770621
0738 IPUT 26622 26642 92002-16006 740801
0739 IABS 26643 26655 750701 24998-16001
0740
0741
0742 MEMORY RESIDENTS
0743
0744 EXTND(0010)30002 30154 92067-16028 REV.1805 771115
0745 RMPAR 30155 30213 771116 24998-16001
0746
0747 D.RTR(0001)30246 32305 92002-16007 1805 780106
0748 P.PAS 32370 32416 92002-16006 740801
0749
0750 PRMPT(0010)32421 33036 92067-16003 REV.1805 780119
0751 TRMLU 33037 33132 92067-16035 REV.1805 771117
0752 IDGET 33133 33215 92067-16037 REV.1805 771227
0753
0754 RSPNS(0010)33220 33735 92067-16003 REV.1805 780119
0755 TRMLU 33742 34035 92067-16035 REV.1805 771117
0756 IDGET 34041 34123 92067-16037 REV.1805 771227
0757
0758 TTYEV(0002)34126 34135 29102-60013
0759
0760 WHZAT(0001)34154 36256 92067-16007 REV.1805 771219
0761
0762 CMM4 (0090)36310 47534
0763 REIO 47540 47644 92067-16035 REV.1805 780221
0764 CLKIU 47645 47653 750701 24998-16001
0765 IAND 47654 47663 750701 24998-16001
0766 IGET 47664 47672 750701 24998-16001
0767 IOR 47673 47702 750701 24998-16001
0768 RMPAR 47703 47741 771116 24998-16001
0769 IXGET 47742 50411
0770 DOIO 50414 51575
0771 DISC3 51611 52340
0772 DTRK 52346 52654
0773
0774
0775
0776 RT DISC RESIDENTS
0777
0778 SMP (0030)26002 31360 92067-16028 REV.1805 771115

```

0779	RNRQ	31361	31622	92067-16035	REV.1805	780222
0780	\$ALRN	31623	31740	92067-16035	REV.1805	770715
0781	PRTN	31744	32056	92067-16035	REV.1805	771005
0782	.DRCT	32063	32071	92067-16035	REV.1805	741120
0783	REIO	32072	32176	92067-16035	REV.1805	780221
0784	READF	32177	32735	92002-16006	770801	
0785	POST	32736	32764	92002-16006	740801	
0786	P.PAS	32765	33013	92002-16006	740801	
0787	RWSUB	33014	33265	92002-16006	750422	
0788	RWNDS	33266	33410	92002-16006	771121	
0789	R/WS	33411	33544	92002-16006	740801	
0790	RMPAR	33545	33603	771116	24998-16001	
0791						
0792	JOB (0030)	26002	27760	92067-16028	REV. 1805	760715
0793	RNRQ	27761	30222	92067-16035	REV.1805	780222
0794	LURQ	30224	30604	92067-16035	REV.1805	771013
0795	\$ALRN	30605	30722	92067-16035	REV.1805	770715
0796	.DRCT	30723	30731	92067-16035	REV.1805	741120
0797	REIO	30732	31036	92067-16035	REV.1805	780221
0798	\$PARS	31037	31257	92067-16035	REV.1805	770621
0799	OPEN	31260	31445	92002-16006	741205	
0800	READF	31472	32230	92002-16006	770801	
0801	CLOSE	32255	32373	92002-16006	771115	
0802	POST	32374	32422	92002-16006	740801	
0803	\$OPEN	32423	32631	92002-16006	740801	
0804	P.PAS	32632	32660	92002-16006	740801	
0805	RWSUB	32661	33132	92002-16006	750422	
0806	RWNDS	33133	33255	92002-16006	771121	
0807	R/WS	33256	33411	92002-16006	740801	
0808	\$OPEN	33412	33462	92002-16006	741025	
0809	RMPAR	33463	33521	771116	24998-16001	
0810						
0811	\$POUT (0011)	26002	26755	92067-16028	REV.1805	780309
0812	LURQ	26756	27336	92067-16035	REV.1805	771013
0813	\$ALRN	27337	27454	92067-16035	REV.1805	770715
0814	.DRCT	27455	27463	92067-16035	REV.1805	741120
0815						
0816						
0817						
0818	CG DISC RESIDENTS					
0819						
0820	\$CNFX (0099)	26002	31460	92067-16006	REV.1805	780112
0821	\$PARS	31461	31701	92067-16035	REV.1805	770621
0822	\$CVT3	31702	31767	92067-16035	REV.1805	770621
0823						
0824	AUTOR (0001)	26002	26361	91730-16009	REV.1805	780203
0825	FIXMP	26362	26444	91730-16008	REV 1805	771206
0826						
0827	LOADR (0080)	26002	41350	92067-16002	REV.1805	780211
0828	LURQ	41351	41731	92067-16035	REV.1805	771013
0829	\$ALRN	41736	42053	92067-16035	REV.1805	770715
0830	PRTN	42054	42166	92067-16035	REV.1805	771005
0831	REIO	42167	42273	92067-16035	REV.1805	780221
0832	IFBRK	42274	42323	92067-16035	REV.1805	770621
0833	\$CVT3	42324	42411	92067-16035	REV.1805	770621
0834	LOGLU	42412	42461	92067-16035	REV.1805	771117
0835	CREAT	42462	42737	92002-16006	741022	
0836	OPEN	42740	43125	92002-16006	741205	
0837	READF	43126	43664	92002-16006	770801	
0838	APUSN	43671	44032	92002-16006	750227	



0839	LDCF	44041	44227	92002-16006	750416
0840	CLOSE	44230	44346	92002-16006	771115
0841	NAM..	44347	44443	92002-16006	740801
0842	\$OPEN	44444	44652	92002-16006	740801
0843	P.PAS	44653	44701	92002-16006	740801
0844	RWSUB	44702	45153	92002-16006	750422
0845	RWNDS	45154	45276	92002-16006	771121
0846	R/WS	45277	45432	92002-16006	740801
0847	NAMR	45433	45727	750701	24998-16001
0848	RMPAR	45730	45766	771116	24998-16001
0849					
0850	GASP (0080)	26002	27415	92067-16028	REV.1805 780317
0851	G1CEX	27416	27527	92002-16001	760615
0852	ST.LU	27530	27705	92067-16028	780317
0853	G1ROT	27715	30070	92002-16001	760615
0854	G0QIP	30073	30360	92002-16001	760621
0855	RNRQ	30361	30622	92067-16035	REV.1805 780222
0856	\$ALRN	30623	30740	92067-16035	REV.1805 770715
0857	.DRCT	30741	30747	92067-16035	REV.1805 741120
0858	REIO	30750	31054	92067-16035	REV.1805 780221
0859	KCVT	31055	31070	92001-16035	REV.1805 770621
0860	PARSE	31071	31110	92067-16035	REV.1805 770714
0861	\$PARS	31111	31331	92067-16035	REV.1805 770621
0862	\$CVT3	31332	31417	92067-16035	REV.1805 770621
0863	OPEN	31420	31605	92002-16006	741205
0864	READF	31623	32361	92002-16006	770801
0865	CLOSE	32407	32525	92002-16006	771115
0866	POST	32526	32554	92002-16006	740801
0867	\$OPEN	32555	32763	92002-16006	740801
0868	P.PAS	32764	33012	92002-16006	740801
0869	RWSUB	33013	33264	92002-16006	750422
0870	RWNDS	33265	33407	92002-16006	771121
0871	R/WS	33410	33543	92002-16006	740801
0872	RMPAR	33544	33602	771116	24998-16001
0873					
0874	GASP1(0099)	33603	33615	92067-16028	REV.1805 760615
0875	G1COJ	33624	34210		
0876	G1CCJ	34223	34637	92002-16001	760615
0877	G1CKS	34640	35454	92002-16001	760627
0878	G1CDS	35505	36532	92002-16001	760621
0879	G1STM	36536	36730	92002-16001	740807
0880	CNUMD	36731	36750	92001-16035	REV.1805 770621
0881					
0882	GASP2(0099)	33603	33613	92067-16028	REV.1805 760615
0883	G1CSD	33624	34242	92002-16001	760622
0884	G1C??	34246	35056	92002-16001	741027
0885	G1CIN	35063	36425	92002-16001	760630
0886	G1CDA	36470	37044	92002-16001	760627
0887	CNUMD	37045	37064	92001-16035	REV.1805 770621
0888	CREAT	37065	37342	92002-16006	741022
0889	PURGE	37343	37441	92002-16006	740801
0890	NAM..	37442	37536	92002-16006	740801
0891					
0892	FMGR (0090)	26002	26757	92002-16008	REV.1805 760627
0893	FM.CH	26760	30767	92002-16008	771208
0894	LURQ	31104	31464	92067-16035	REV.1805 771013
0895	\$ALRN	31465	31602	92067-16035	REV.1805 770715
0896	.DRCT	31603	31611	92067-16035	REV.1805 741120
0897	IFBRK	31612	31641	92067-16035	REV.1805 770621
0898	IFTTY	31642	31715	92067-16035	REV.1805 771208

0899	OPEN	31724	32111	92002-16006	741205
0900	CLOSE	32124	32242	92002-16006	771115
0901	\$OPEN	32243	32451	92002-16006	740801
0902	RWNDS	32452	32574	92002-16006	771121
0903	R/WS	32575	32730	92002-16006	740801
0904	RMPAR	32731	32767	771116	24998-16001
0905					
0906	FMGR0(0099)	32770	32775	92002-16008	740801
0907	PK..	33006	34431		
0908	CR..	34527	35611	92002-16008	760616
0909	COR..A	35612	35632	92067-16035	REV.1805 770621
0910	READF	35647	36405	92002-16006	770801
0911	REIO	36434	36540	92067-16035	REV.1805 780221
0912	RWNDF	36541	36622	92002-16006	740801
0913	NAM..	36623	36717	92002-16006	740801
0914	P.PAS	36720	36746	92002-16006	740801
0915	RWSUB	36747	37220	92002-16006	750422
0916	LOCK.	37221	37270	92002-16006	771118
0917	FM.UT	37271	40436	92002-16006	771118
0918	CREA.	40502	40553		
0919	CREAT	40554	41031	92002-16006	741022
0920					
0921	FMGR1(0099)	32770	33116	92002-16008	760929
0922	.PARS	33120	34403	92002-16008	765025
0923	C.TAB	34472	34635	92002-16008	760720
0924	CA..	34636	35057	92002-16008	760513
0925	REA..C	35060	35132	92002-16008	770823
0926	EE..	35133	35173	92002-16008	760512
0927	TR..	35174	35425	92002-16008	760616
0928	MR..	35426	35670	92002-16008	760621
0929	SE..	35672	36056		
0930	IF..	36072	36307	92002-16008	760929
0931	AB..	36310	36536	92002-16008	780221
0932	OP..	36537	36604	92002-16008	760511
0933	MESSS	36605	36744	92067-16035	REV.1805 771227
0934	CNUMD	36745	36764	92001-16035	REV.1805 770621
0935	\$CVT3	36765	37052	92067-16035	REV.1805 770621
0936	READF	37053	37611	92002-16006	770801
0937	REIO	37612	37716	92067-16035	REV.1805 780221
0938	POSNT	37720	40163	92002-16006	760702
0939	P.PAS	40207	40235	92002-16006	740801
0940	RWSUB	40236	40507	92002-16006	750422
0941	WRLG.	40510	40657	92002-16006	760622
0942	CK.SM	40660	41003	92002-16006	REV. 1805 771205
0943					
0944	FMGR2(0099)	32770	33000	92002-16008	760622
0945	IN.IT	33004	34101	92002-16008	780106
0946	IN..	34120	36030	92002-16008	771229
0947	MC..	36043	36361	92002-16008	760511
0948	RC..	36362	36547		
0949	PU..	36550	36772		
0950	PURGE	36773	37071	92002-16006	740801
0951	NAM..	37072	37166	92002-16006	740801
0952	J.PUT	37167	37213	92002-16006	740801
0953	IPUT	37214	37234	92002-16006	740801
0954	FID.	37235	37354		
0955	MSC.	37355	37411		
0956	LOCK.	37412	37461	92002-16006	771118
0957	FM.UT	37462	40627	92002-16006	771118
0958	.DPSY	40630	40632	771116	24998-16001

0959					
0960	FMGR3(0099)	32770	32775	92002-16008	760720
0961	DL..	33004	34301	92002-16008	771020
0962	F.SET	34361	34551	92002-16006	760719
0963	CS..	34552	35000	92002-16008	760318
0964	READF	35001	35537	92002-16006	770801
0965	REIO	35540	35644	92067-16035	REV.1805 780221
0966	LOCF	35654	36042	92002-16006	750416
0967	P.PAS	36060	36106	92002-16006	740801
0968	RWSUB	36107	36360	92002-16006	750422
0969	MSC.	36361	36415		
0970	FM.UT	36416	37563	92002-16006	771118
0971	CK.ID	37564	37606	92002-16006	771205
0972	LULU.	37607	37677	92002-16006	760227
0973					
0974	FMGR4(0099)	32770	33001	92002-16008	760622
0975	ST.DU	33003	34254	92002-16008	760622
0976	CO..	34310	35012		
0977	F.UTM	35013	35254	92002-16008	760514
0978	CREAT	35255	35532	92002-16006	741022
0979	READF	35554	36312	92002-16006	770801
0980	REIO	36333	36437	92067-16035	REV.1805 780221
0981	RWNDF	36440	36521	92002-16006	740801
0982	LOCF	36522	36710	92002-16006	750416
0983	NAM..	36711	37005	92002-16006	740801
0984	P.PAS	37006	37034	92002-16006	740801
0985	RWSUB	37035	37306	92002-16006	750422
0986	FM.UT	37307	40454	92002-16006	771118
0987	CREA.	40525	40576		
0988	CK.SM	40577	40722	92002-16006	REV. 1805 771205
0989					
0990	FMGR5(0099)	32770	33002	92002-16008	760622
0991	RU..	33003	33641	92002-16008	761004
0992	RP..	33642	33751	92002-16008	761004
0993	SESSN	33754	34012	92002-16008	761005
0994	.RENM	34015	34141	92002-16008	761004
0995	.EXCP	34142	34206	92002-16008	761002
0996	IDDUP	34207	34551	92002-16008	770902
0997	IDKPL	34552	35277	92002-16008	780106
0998	IDKPD	35300	35517	92002-16008	771115
0999	OPMES	35520	35710	92002-16008	760513
1000	TL..	35711	35730		
1001	MESSS	35735	36074	92067-16035	REV.1805 771227
1002	READF	36102	36640	92002-16006	770801
1003	REIO	36641	36745	92067-16035	REV.1805 780221
1004	NAM..	36746	37042	92002-16006	740801
1005	P.PAS	37043	37071	92002-16006	740801
1006	RWSUB	37072	37343	92002-16006	750422
1007	ID.A	37344	37433	92002-16008	780207
1008	CNT.	37434	37670	92002-16006	760520
1009	FCONT	37671	37766	92002-16006	751104
1010	HUMP.	37770	40026	92002-16006	741025
1011	SET.T	40033	40061	92002-16006	740801
1012	TL.	40062	40115	92002-16006	760322
1013	ST.TM	40116	40152	92002-16006	741223
1014					
1015	FMGR6(0099)	32770	33000	92002-16008	740801
1016	CN..	33001	33041		
1017	JO..	33046	34050	92002-16008	760719
1018	EO..	34063	34666	92002-16008	770620

1019	OF..	34667	34762	92002-16008	740820	
1020	LG..	34763	35010	92002-16008	760517	
1021	RNRQ	35011	35252	92067-16035	REV.1805	780222
1022	KCVT	35253	35266	92001-16035	REV.1805	770621
1023	MESSS	35267	35426	92067-16035	REV.1805	771227
1024	SCVT3	35427	35514	92067-16035	REV.1805	770621
1025	NAMF	35515	35670	92002-16006	771115	
1026	READF	35701	36437	92002-16006	770801	
1027	REIO	36471	36575	92067-16035	REV.1805	780221
1028	POST	36576	36624	92002-16006	740801	
1029	NAM..	36625	36721	92002-16006	740801	
1030	P.PAS	36722	36750	92002-16006	740801	
1031	RWSUB	36751	37222	92002-16006	750422	
1032	SPOPN	37223	37273	92002-16006	741025	
1033	SET.T	37274	37322	92002-16006	740801	
1034	ST.TM	37323	37357	92002-16006	741223	
1035	B.FLG	37360	37426	92002-16006	741118	
1036	LULU.	37427	37517	92002-16006	760227	
1037	RANGE	37520	37543	92002-16006	740801	
1038	ONOFF	37566	40131	92002-16006	750128	
1039	EX.TM	40133	40350	92002-16006	771115	
1040	IPUT	40351	40371	92002-16006	740801	
1041	LU.CL	40372	40440	92002-16006	760702	
1042	AVAIL	40441	40533	92002-16006	741231	
1043						
1044	FMGR7(0099)	32770	32776	92002-16008	760702	
1045	??..	33000	35355	92002-16008	771111	
1046	SY..	35356	35414	92002-16008	760520	
1047	NX.JB	35417	36311	92002-16008	760702	
1048	RNRQ	36364	36625	92067-16035	REV.1805	780222
1049	MESSS	36626	36765	92067-16035	REV.1805	771227
1050	READF	36766	37524	92002-16006	770801	
1051	REIO	37525	37631	92067-16035	REV.1805	780221
1052	POST	37632	37660	92002-16006	740801	
1053	P.PAS	37661	37707	92002-16006	740801	
1054	RWSUB	37717	40170	92002-16006	750422	
1055	SPOPN	40173	40243	92002-16006	741025	
1056	B.FLG	40244	40312	92002-16006	741118	
1057	LULU.	40313	40403	92002-16006	760227	
1058	LII.CL	40404	40452	92002-16006	760702	
1059						
1060	FMGR8(0099)	32770	32776	92002-16008	740801	
1061	SA..	32777	33751	92002-16008	760621	
1062	SP..	33752	34721			780221
1063	MS..	34753	35246			
1064	PRIN	35247	35361	92067-16035	REV.1805	771005
1065	READF	35402	36140	92002-16006	770801	
1066	REIO	36157	36263	92067-16035	REV.1805	780221
1067	RWNDF	36264	36345	92002-16006	740801	
1068	LOCF	36346	36534	92002-16006	750416	
1069	P.PAS	36535	36563	92002-16006	740801	
1070	RWSUB	36564	37035	92002-16006	750422	
1071	IPUT	37036	37056	92002-16006	740801	
1072	CREA.	37057	37130			
1073	CREAT	37131	37406	92002-16006	741022	
1074	NAM..	37407	37503	92002-16006	740801	
1075	CK.SM	37504	37627	92002-16006	REV.1805	771205
1076	ID.A	37630	37717	92002-16008	780207	
1077	WRISS	37720	37756	92002-16006	740801	
1078	READ.	37757	40003	92002-16006	740801	

1079	XWRIS	40006	40403	750701	24998-16001
1080	SREAD	40404	41046	771116	24998-16001
1081					
1082	FMGR9(0099)	32770	32776	92002-16008	760720
1083	LI..	33001	34503	92002-16008	760720
1084	CL..	34603	35064		
1085	LU..	35071	36212	92002-16008	760702
1086	RNRQ	36253	36514	92067-16035	REV.1805 780222
1087	KCVT	36515	36530	92001-16035	REV.1805 770621
1088	SCVT3	36531	36616	92067-16035	REV.1805 770621
1089	READF	36617	37355	92002-16006	770801
1090	REIO	37356	37462	92067-16035	REV.1805 780221
1091	FSTAT	37463	37507	92002-16006	740801
1092	LOCF	37510	37676	92002-16006	750416
1093	POST	37677	37725	92002-16006	740801
1094	P.PAS	37726	37754	92002-16006	740801
1095	RWSUB	37757	40230	92002-16006	750422
1096	SPUPN	40232	40302	92002-16006	741025
1097	LULU.	40303	40373	92002-16006	760227
1098	RANGE	40374	40417	92002-16006	740801
1099	AVAIL	40420	40512	92002-16006	741231
1100					
1101	LUMAP(0020)	26002	27065		
1102	FMTIO	27074	30512	24998-16002	REV.1805 780303
1103	REIO	30560	30664	92067-16035	REV.1805 780221
1104	FMT.E	30665	30665	24998-16002	REV.1805 780303
1105	FRMTR	30726	33527	24998-16002	REV.1805 780303
1106	CLKIO	33740	33746	750701	24998-16001
1107	RMPAR	33750	34006	771116	24998-16001
1108	PNAME	34010	34055	771121	24998-16001
1109					
1110	KYDMP(0010)	26002	27326		
1111	KCVT	27327	27342	92001-16035	REV.1805 770621
1112	SCVT3	27343	27430	92067-16035	REV.1805 770621
1113	OPEN	27431	27616	92002-16006	741205
1114	READF	27634	30372	92002-16006	770801
1115	REIO	30422	30526	92067-16035	REV.1805 780221
1116	CLOSE	30527	30645	92002-16006	771115
1117	SOOPEN	30646	31054	92002-16006	740801
1118	P.PAS	31055	31103	92002-16006	740801
1119	RWSUB	31104	31355	92002-16006	750422
1120	RWNDS	31356	31500	92002-16006	771121
1121	R/W*	31501	31634	92002-16006	740801
1122	CLKIO	31635	31643	750701	24998-16001
1123	IAND	31644	31653	750701	24998-16001
1124	RMPAR	31654	31712	771116	24998-16001
1125					
1126	USPMP(0099)	26002	26437	91730-16003	REV.1805 780117
1127	FMTIO	26442	30060	24998-16002	REV.1805 780303
1128	REIO	30064	30170	92067-16035	REV.1805 780221
1129	FMT.E	30171	30171	24998-16002	REV.1805 780303
1130	FRMTR	30212	33013	24998-16002	REV.1805 780303
1131	CLKIO	33154	33162	750701	24998-16001
1132	RMPAR	33163	33221	771116	24998-16001
1133	PNAME	33222	33267	771121	24998-16001
1134	CNVSC	33270	33334	91730-16004	REV.1805 771219
1135	REPT	33365	35233		
1136					
1137	EXMP (0099)	26002	32006	91730-16002	REV.1805 780117
1138	FMTIO	32007	33425	24998-16002	REV.1805 780303

```

1139      REID      33426 33532 92067-16035 REV.1805 780221
1140      FMT,E     33533 33533 24998-16002 REV.1805 780303
1141      FRMTR     33560 36361 24998-16002 REV.1805 780303
1142      CLRIO     36401 36407 750701 24998-16001
1143      PAUSE     36410 36510 771122 24998-16001
1144      RMPAR     36511 36547 771116 24998-16001
1145      PAU,E     36550 36550 750701 24998-16001
1146      PNAME     36551 36616 771121 24998-16001
1147
1148      LGTAT(0099)26002 30050 92067-16008 REV.1805 780127
1149      SCVT3      30051 30136 92067-16035 REV.1805 770621
1150
1151
1152
1153
1154      RT PARTITION REQMTS:
1155      SMP      04 PAGES
1156      JOB      04 PAGES
1157      SPOUT    02 PAGES
1158
1159      BG PARTITION REQMTS:
1160      SCNFX    03 PAGES
1161      AUTOR    02 PAGES
1162      LOADR    09 PAGES
1163      GASP     06 PAGES
1164      FMGR     07 PAGES
1165      LUMAP    05 PAGES
1166      KYDMP    03 PAGES
1167      DSPMP    05 PAGES
1168      EXMP     06 PAGES
1169      LGTAT    03 PAGES
1170
1171      MAXIMUM PROGRAM SIZE:
1172      W/O COM 29 PAGES
1173      W/  COM 28 PAGES
1174      W/  TA2 22 PAGES
1175
1176
1177      SYS AV MEM: 02944 WORDS
1178
1179      1ST PART PG 00044
1180      CHANGE 1ST PART PG ?
1181      TR
1182      CHANGE 1ST PART PG ?
1183      45
1184
1185      SYS AV MEM: 03968 WORDS
1186
1187      PAGES REMAINING: 00083
1188
1189      DEFINE PARTITIONS:
1190
1191      PART 01?
1192      2,BG
1193
1194      PART 02?
1195      5,BG
1196
1197      PART 03?
1198      11,BG

```

1199  
 1200 PART 04?  
 1201 65,BG  
 1202  
 1203 SUBPARTITIONS?  
 1204 YES  
 1205  
 1206 PART 05?  
 1207 7,S  
 1208  
 1209 PART 06?  
 1210 7,S  
 1211  
 1212 PART 07?  
 1213 22,S  
 1214  
 1215 PART 08?  
 1216 29,S  
 1217  
 1218 PART 09?  
 1219 /E  
 1220  
 1221 MODIFY PROGRAM PAGE REQUIREMENTS?  
 1222 -  
 1223 FMGR,7  
 1224 -  
 1225 LOADK,20  
 1226 -  
 1227 /E  
 1228  
 1229 ASSIGN PROGRAM PARTITIONS?  
 1230 -  
 1231 /E  
 1232  
 1233 SYSTEM STORED ON DISC  
 1234 SYS SIZE: 35 TRKS, 036 SECS  
 1235  
 1236 RT4GN FINISHED  
 1237  
 1238 \*\*\*\*\* ERRORS

# PARTITION STATES

```
000
000 0:11:43:730
000*****
000PTN#  SIZE    PAGES  BG/RT PRGRM
000*****
000
000
000
000
000
000
000 7S      22      77-   98 BG    FMG16
000
000
0001
00011 <UNDEFINED>
00012 <UNDEFINED>
00013 <UNDEFINED>
00014 <UNDEFINED>
00015 <UNDEFINED>
000*****
000 0:11:43:780
000
```

NOTE: \$MATA= 23500B  
\$MNP = 15.



# PROGRAM STATES

```
000
000 0111:28:690
000*****
\00PT SZ PRGRM,T ,PRIOR*DRMT*SCHD*I/O *WAIT*MEMY*DISC*OPER * NEXT TIME *
000*****
000
000
000 2 5 TVST *3 *00001 0 ***** 0111:28:950
000
000
000
000*****
000DOWN LU'S,
000*****
000DOWN EQT'S
000*****
000 0111:28:730
000
```

## LOCATIONS 1600 THROUGH

1677

033127	037624	035725	033016	032253	025467	024715	024352*	6W?	1	6	4	+7)	(
024325	014011	114016	014016	014015	003251	003263	003277*	(					
003232	012000	003214	003207	006057	006274	006120	006135*					/	P)
006067	006302	006303	006207	006255	006240	006307	006266*	7					
006265	002000	003221	003313	003223	000000	000000	036306*						<
002023	000023	002762	000051	003104	000072	023717	016071*					)	D : ' 9
002136	002137	002140	002141	002142	002143	002144	002145*	↑	+				
002146	002147	002150	000006	000015	002042	000003	051474*					"	S<

## LOCATIONS 1700 THROUGH

1777

000003	051552	051540	000000	000000	000000	000000	000000*	S	S				
000000	016535	000000	017755	000000	000000	000000	016535*	)					)
016535	016536	016537	016540	016541	016542	016543	016544*	)	↑	+			
016545	016546	016547	016550	000000	000000	031017	000000*					2	
000224	000057	000002	001445	000002	026000	011107	000144*	/	X	,	G		
052654	052654	011253	000525	052654	177400	000400	000140*	U	UU				
000000	010422	001224	004646	000234	000000	000000	000000*						
000000	002151	002152	002153	002154	030000	000000	052654*					U	

## LOCATIONS 2000 THROUGH

2077

177772	000000	011000	000400	000403	011000	000226	000000*						
020000	000313	000147	020000	000313	000316	020000	000313*						
000465	020000	000313	000000	005127	004214	100017	015120*	5		W		P	
001101	036575	177700	000042	000105	136575	000100	000000*	A	=	"	E		
000000	000000	000000	004056	004145	010024	002400	000401*			.			
003504	000021	007230	007252	005752	000401	002460	100000*	D				0	
000000	000000	004016	005106	100010	017000	000000	000000*			F			
000000	000000	000000	000000	000000	000000	000000	000000*						

## LOCATIONS 2100 THROUGH

2177

000000	004065	004601	000023	017400	000000	000000	000000*	5					
000000	000000	000000	000031	002475	164217	000000	017714*						
004056	004145	010025	102402	000401	003431	000044	007062*	.				S	2
007172	004517	000401	002526	100000	101135	061240	005553*	0		V		)	
006100	040016	105000	040202	142542	177666	177711	000000*	00		0			
100047	177754	177703	100000	100001	000000	006763	007200*	1					
040025	005000	000000	000000	000000	000000	000000	000000*						
000000	000000	100000	000000	000000	006322	007137	100021*					+	

## LOCATIONS 2200 THROUGH

2277

011401	000403	177776	000044	000000	177766	000006	000000*			S			
130740	100000	000000	000000	004070	004637	000022	003400*			8			
000000	000000	000000	000000	000000	000000	000005	002543*						
000000	000000	000000	004070	004637	000077	003400	000000*			8		?	
000000	000000	000000	000000	000000	000005	002550	000000*						
000000	000000	004070	004637	000077	003400	000000	000000*			8		?	
000000	000000	000000	000000	000005	002555	000000	000000*						
000000	007166	007401	130120	005400	000002	033226	000036*			P		6	

## LOCATIONS 2300 THROUGH

2377

000000	000000	000000	000000	000000	177753	000000	000000*						
004070	004637	000077	003400	000000	000000	000000	000000*	8		?			
000000	000000	000005	002562	000000	000000	000000	004056*						.
004145	040026	002400	000000	000000	000000	000000	000000*						
000000	000015	002567	100000	000000	000000	012654	014266*						
000060	021400	000000	000000	000000	000000	000000	000000*	00					
000022	002604	000000	000000	000000	012654	014266	000061*					1	
021400	000000	000000	000000	000000	000000	000000	000022*						

## LOCATIONS 2400 THROUGH

2477

002626	000000	000000	000000	012654	014266	000062	021400*					20	
000000	000000	000000	000000	000000	000000	000022	002650*						
000000	000000	000000	012654	014266	000063	021400	000000*					30	
000000	000000	000000	000000	000000	000022	002672	000000*						
000000	000000	012577	012472	000004	021400	000000	000000*						H-28

000000	000000	000000	000000	000000	000000	000000	000000*
000001	016351	000000	000000	000020	000000	000000	006216*
000000	000000	000000	006423	000000	000000	000000	000000*
LOCATIONS 2500 THROUGH				2577			
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000001	016351*
000000	000000	000000	000000	000000	006216	000000	000000*
000000	006423	000002	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
LOCATIONS 2600 THROUGH				2677			
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
LOCATIONS 2700 THROUGH				2777			
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000002	000030	000032	000036*
000030	000002	000030	000002	000034	000034	000034	000002*
000034	000030	100001	100001	100001	100001	100001	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000002	000001	000000	004002	010002	000006*
014002	000110	000011	000012	000013	004014	000015	004001*
LOCATIONS 3000 THROUGH				3077			
000000	000005	000005	000005	000005	000000	010001	000004*
004004	010004	014004	014001	000016	004016	010016	014016*
020001	000003	004003	000017	000020	000021	000022	000006*
000007	024001	000023	000000	000000	000000	000000	000000*
000000	000000	100000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*
000000	000000	000000	000000	000000	000000	000000	000000*

H

(

## LOCATIONS 16000 THROUGH 16077

016030	016033	016036	016041	016044	016047	016052	016055*	1 5 1 *
016060	016063	016066	000000	000000	000000	000000	000000*	0 3 6
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	016257	016314	016351	016406	016443	016500	016535*	* 0 J

## LOCATIONS 16100 THROUGH 16177

016572	016633	016674	016735	016776	017037	017100	017141*	0
017202	017243	017304	017345	017406	017447	017510	017551*	1 H
017612	017653	017714	017755	020016	020057	020120	020161*	/ P
020222	020263	020324	020365	020426	020467	020530	020571*	1 171X1
020632	020673	020734	020775	021036	021077	021140	021201*!	! ! ! " "2" "
021242	021303	021344	021405	021446	021507	021550	021611*"	" " # #G# #
021652	021713	021754	022015	022056	022117	022160	022221*#	# # \$ \$ \$ \$ \$ \$
022262	022323	022364	022425	022466	022527	022570	022616*#	\$ \$ X X6X#X X

## LOCATIONS 16200 THROUGH 16277

022627	022640	022651	022662	022673	022704	022715	022726*#	X X X X X X X
022737	022750	022761	022772	023003	023014	023025	023036*#	X X X X X X X
023047	023060	023071	023102	023113	023124	023135	023146*#	'A0A9B&K&T&J&
023157	023170	023201	023212	023223	023234	023245	023256*#	& & & & & &
023267	023300	023311	023322	023333	023344	023355	023366*#	& & & & & &
023377	023410	023421	023432	023443	023454	000000	000000*#	! ! ! ! # ! ,
000000	000000	000000	000000	000000	000012	030011	000000*	0
000000	016260	030016	042530	052116	042001	000000	000000*	0 EXTND

## LOCATIONS 16300 THROUGH 16377

000000	025000	177574	000000	001000	030000	030214	000004*	* 0 0
000011	000000	000000	000000	017612	000000	000000	000000*	
000000	000000	000001	030460	000000	002301	016315	031010*	10 2
042056	051124	051001	000000	000000	000000	025000	177574*0.RTR	*
000000	000200	030214	032417	000011	000013	000000	000000*	0 5
000000	017612	002003	000000	177767	000000	000000	000012*	
032421	032737	002402	000000	032732	050122	046520	052001*5 5	5 PKMPT
000000	000000	000000	025000	177574	000000	000200	032417*	* 5

## LOCATIONS 16400 THROUGH 16477

033216	000013	000016	000000	000000	000000	000000	000000*6	
000000	000000	000000	000000	000012	033220	000000	000000*	6
016407	000000	051044	050116	022001	000000	000000	000000*	RSPNS
025000	177574	000000	000200	033216	034124	000016	000024*#	6 BT
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000002	034126	000000	000000	016444	000000	052124*	BV \$ T1
054505	053001	000000	000000	000000	025000	177574	000000*YEV	*
001000	034124	034136	000024	000026	000000	000000	000000*	BTBT

## LOCATIONS 16500 THROUGH 16577

017612	000000	000000	000001	000000	000000	000001	034154*	8
000000	002400	016501	135336	053510	055101	052001	000000*	A WHZAT
000000	000000	025000	177574	000000	000200	034136	036306*	* BT<
000026	000031	000000	000000	000000	017612	000054	103324*	,
000021	000000	000000	000132	041015	050017	051466	050014*	ZB P S6P
100000	041515	046464	020001	000001	000000	000000	025000*	CHM4 *
177574	040017	000200	036306	052655	000031	000266	000000*	0 < U
000000	177777	000000	000000	000000	000000	000000	000000*	

## LOCATIONS 16600 THROUGH 16677

000036	026064	000000	000000	016573	000000	051515	050040*	,4 SMP
020002	000000	000000	000000	025000	177574	000000	007000*	*
026000	033604	000002	000173	001616	000000	000000	000000*	, 7
000000	000000	000000	000000	000000	000000	000000	000000*	
000000	000036	026002	000000	000000	016634	000000	045117*	H-30 JU

041040	020002	000000	000000	000000	025000	177574	000000*	B		*
007000	026000	033522	000002	000045	001700	000000	000000*		, 7R	X
000000	000000	000000	000000	000000	000000	000000	000000*			
LOCATIONS 16700 THROUGH 16777										
000000	000000	000013	026107	000000	000000	016675	000000*		,G	
051520	047525	052002	000000	000000	000000	025000	177574*	SPOUT		*
000000	003000	026000	027464	000002	000021	002020	000000*		, /4	
000000	000000	000000	000000	000000	000000	000000	000000*			
000000	000000	000000	000143	026273	000000	000000	016736*			
000000	026054	026054	026003	000000	000000	020000	025000*		, , , ,	*
177574	000000	005200	026000	031770	000002	000162	002040*		, 3	
000000	000000	000000	000000	000000	000000	000000	000000*			
LOCATIONS 17000 THROUGH 17077										
000000	000000	000000	000000	000001	026002	000000	000000*			
016777	000000	040525	052117	051003	000000	000000	000000*	AUTOR		
025000	177574	000000	003000	026000	026445	000002	000004*		, -X	
002102	000000	000000	000000	000000	000000	000000	000000*	B		
000000	000000	000000	000000	000000	000120	026104	000000*		P, D	
000000	017040	000000	046117	040504	051003	000000	000000*	LOADR		
000000	025000	177574	000000	047200	026000	045767	000002*	*	N , K	
000651	002112	000000	000000	000000	000000	000000	000000*	J		
LOCATIONS 17100 THROUGH 17177										
000000	000000	000000	000000	000000	000120	026661*			P=	
000000	177777	017101	127064	043501	051520	020003	000000*	A 4GASP		
000000	000000	025000	177574	000000	013005	026000	033603*	*	, 7	
000002	000064	002422	000000	000000	037537	000000	000000*	4	?+	
000000	000000	016535	103577	000044	000000	000000	000132*	J	\$ Z	
026222	033312	000000	177777	133276	043115	043522	020043*	, 6	FMGR	*
110003	000000	000000	025000	177574	000000	015202	026000*	*		
032770	000002	000045	002642	000000	000000	041047	000000*5	X	B'	
LOCATIONS 17200 THROUGH 17277										
000000	177777	017612	000601	103646	177572	000000	000000*			
000024	026144	026245	000007	027271	126240	046125	046501*	, ,	LUMA	
050003	000000	000000	000000	025000	177574	000000	007205*	*		
026000	033675	000002	000354	016400	000000	000000	000000*	, 7		
000000	000000	000000	000000	000000	000000	000000	000000*			
000000	000012	026002	000000	000000	017244	000000	045531*	,	KY	
042115	050003	000000	000000	000000	025000	177574	000000*DMP	*		
005200	026000	031713	000002	000025	004246	000000	000000*	, 3		
LOCATIONS 17300 THROUGH 17377										
000000	000000	000000	000000	000000	000000	000000	000000*			
000000	000000	000143	026012	000000	000000	017305	000000*	,		
042123	050115	050003	000000	000000	000000	025000	177574*DSPMP	*		
000000	011200	026000	035237	000002	000027	004310	000000*	, :		
000000	000000	000000	000000	000000	000000	000000	000000*			
000000	000000	000000	000143	031154	000000	000000	017346*	2		
000000	042530	046520	020003	000000	000000	000000	025000*	EXMP	*	
177574	000000	013200	026000	036617	000002	000253	004446*	, =		
LOCATIONS 17400 THROUGH 17477										
000000	000000	000000	000000	000000	000000	000000	000000*			
000000	000000	000000	000000	000143	026002	000000	000000*			
017407	000000	046107	052101	052003	000000	000000	000000*	LGTAT		
025000	177574	000000	005200	026000	030137	000002	000042*	, 0+	"	
004622	000000	000000	000000	000000	000000	000000	000000*			
000000	000000	000000	000000	000000	000143	026002	000000*			
000000	017450	000000	054122	042506	020203	000000	000000*	( XREF		
000000	025000	177574	000000	035200	026000	041173	000002*	*	, B	
LOCATIONS 17500 THROUGH 17577										
000436	026222	000000	000000	000000	000000	000000	000000*	,		
000000	000000	000000	000000	000000	000000	000143	026046*		, 8	

000000	002400	017511	145422	046511	041522	047603	000000*	I	MICRO
000000	000000	025000	177574	000000	017200	026000	042176*	*	, U
000002	000511	067474	000000	000000	000000	000000	000000*	I	<
000000	017612	000006	033226	000035	000000	000000	000001*	6	
026720	031333	026476	000000	131324	052126	051524	020303*	2 ->	TVST
020000	000000	050000	007466	177575	000000	011001	026000*	P 6	,

LOCATIONS 17600 THROUGH 17677

034320	000002	000151	063100	000000	000000	000000	000000*		
000000	000000	000000	000001	000000	000000	000000	000000*		
077777	026003	026017	000000	017613	003116	046505	046440*	, ,	NMEM
020303	000001	000000	000000	025000	177574	000000	003000*		*
026000	026037	000002	000006	063072	000000	000000	000000*	, ,	:
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		

LOCATIONS 17700 THROUGH 17777

000000	000000	000200	000000	000000	004244	000000	000000*		
000000	000000	000000	000000	000000	000401	103564	000044*		\$
000000	000000	000132	026222	037706	037677	000047	137676*	Z, ? ?	!
043115	043461	033243	100002	000000	000000	025000	177574*FMG16		*
026003	015206	026000	032770	000002	000045	025204	000000*,	, 5	X*
000000	041047	000000	000000	177760	017141	000004	177776*	B'	
000044	000000	000000	000132	026222	037745	037745	000010*	\$	Z, ? ?
137744	043115	043460	032643	100003	000000	000000	025000*	FMG05	*

LOCATIONS 20000 THROUGH 20077

177574	000400	015204	026000	032770	000002	000045	025204*	, 5	X*
000000	000000	041047	000000	000000	177777	017612	000000*	B'	
000000	000000	000000	000000	000132	026007	000000	002402*		Z,
020017	126022	000000	000000	000000	000000	000000	000000*		
025000	177574	000000	003207	026000	026761	000002	000011*		-
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		

LOCATIONS 20100 THROUGH 20177

000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		

LOCATIONS 20200 THROUGH 20277

000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		

LOCATIONS 20300 THROUGH 20377

000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		
000000	000000	000000	000000	000000	000000	000000	000000*		

[illegible][illegible][illegible][illegible][illegible][illegible]

000000	000000	000000	000000	000000	000000	000000
000000	000000	000000	000000	000000	000000	000000★
000000	000000	000000	000000	000000	000000	000000★
000000	000000	000000	000000	000000	000000	000000★
000000	000000	000000	000000	000000	000000	000000★
000000	000000	000000	000000	000000	000000	000000★



H-34



000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*

LOCATIONS 22200 THROUGH 22277

000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*

LOCATIONS 22300 THROUGH 22377

000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*

LOCATIONS 22400 THROUGH 22477

000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*

LOCATIONS 22500 THROUGH 22577

000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*

LOCATIONS 22600 THROUGH 22677

000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 000000 000000 000000 000000 000000 000000 000000\*  
000000 033603 043501 051520 030425 033603 036751 000064\*  
000122 002504 033603 043501 051520 031025 033603 037537\* R D7 GASP2 7 ?+  
000064 000122 002600 032770 043115 043522 030025 032770\* 4 R 5 FMGR0 5  
041032 000045 000105 002714 032770 043115 043522 030425\*B X E 5 FMGR1  
032770 041004 000045 000147 003040 032770 043115 043522\*5 B X 5 FMGR

LOCATIONS 22700 THROUGH 22777

031025 032770 040633 000045 000113 003124 032770 043115\*2 5 A X K T5 FM  
043522 031425 032770 037700 000045 000104 003246 032770\*GR3 5 ? X D 5  
043115 043522 032025 032770 040723 000045 000112 003320\*FMGR4 5 A X J  
032770 043115 043522 032425 032770 040153 000045 000130\*5 FMGR5 5 0 X X  
003442 032770 043115 043522 033025 032770 040534 000045\* "5 FMGR6 5 A\ X  
000143 003516 032770 043115 043522 033425 032770 040453\* N5 FMGR7 5 A+  
000045 000122 003636 032770 043115 043522 034025 032770\* X R 5 FMGR8 5  
041047 000045 000113 003716 032770 043115 043522 034425\*B\ X K 5 FMGR9

LOCATIONS 23000 THROUGH 23077

032770	040513	000045	000120	004042	044622	043064	027060*5	AK X P "I F4.0
020225	044621	050405	000613	000752	026666	045014	043064*	I Q - J F4
027061	020225	044621	051044	000613	000761	026730	044622*.1	I RS - J
043064	027062	020225	044621	045573	000613	000627	027040*F4.2	I K .
044655	043064	027063	020225	044621	047016	000613	000641*I F4.3	I N
027054	044622	043064	027064	020225	044621	051162	000613*.,I F4.4	I R
000753	027102	044622	043064	027065	020225	044621	050507* .BI F4.5	I RG
000613	000722	027212	040470	040523	046502	030225	040446*	. ABASMB0 A8

LOCATIONS 23100 THROUGH 23177

042327	000415	000470	026026	040527	040523	046502	030625*0	B, AWASMB1
040446	042522	000415	000540	026052	040524	040523	046502*ABER	,*ATASMB
031225	040446	042555	000415	000512	026100	040510	040523*2 ARE	J,0AHAS
046502	031625	040446	041365	000415	000432	026126	040513*MB3 A&B	,VAK
040523	046502	032225	040446	041774	000415	000430	026202*ASMB4 A&C	,
032671	041101	051503	030625	032263	035750	000123	000357*5 BASC1 4 ? S	
034060	032367	041101	051503	031225	032254	037777	000123*004 BASC2 4 ? S	
000176	034124	032422	041101	051503	031625	032264	035030* BT5 BASC3 4 :	

LOCATIONS 23200 THROUGH 23277

000123	000240	034246	032551	041101	051503	032225	032266* S	0 5 BASC4 4
041664	000123	000527	034300	032475	041101	051503	032625*C	S W8 5=BASC5
032273	036630	000123	000313	034444	032434	041101	051503*4 =	S 955 BASC
033225	032273	036706	000123	000364	034514	032564	041101*6 4 =	S 9L5 BA
051503	033625	032306	036332	000123	000313	034630	032370*SC7 4 <	S 9 4
041101	051503	034225	032265	034163	000123	000205	034676*BASCA 4 0	S 9
000000	000000	000000	000020	000000	000000	000000	000000*	
000000	000000	000000	000000	000020	000000	000000	000000*	

LOCATIONS 23300 THROUGH 23377

000000	000000	000000	000000	000000	000020	000000	000000*	
000000	000000	000000	000000	000000	000000	000020	000000*	
000000	000000	000000	000000	000000	000000	000000	000020*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000020	000000	000000	000000	000000	000000	000000	000000*	
000000	000020	000000	000000	000000	000000	000000	000000*	
000000	000000	000020	000000	000000	000000	000000	000000*	
000000	000000	000000	000020	000000	000000	000000	000000*	

LOCATIONS 23400 THROUGH 23477

000000	000000	000000	000000	000020	000000	000000	000000*	
000000	000000	000000	000000	000000	000020	000000	000000*	
000000	000000	000000	000000	000000	000020	000000	000000*	
000000	000000	000000	000000	000000	000000	000000	000020*	
000000	000000	000000	000000	000000	000000	000000	000000*	
000020	000000	000000	000000	000000	000000	000000	000000*	
000000	000020	000000	000000	000000	000000	000000	000000*	
000000	000000	000020	000000	000000	000000	000000	000000*	

LOCATIONS 23500 THROUGH 23577

023534	077777	017612	000055	000001	000001	000000	000000*	'\
000001	017551	000057	000004	000001	000000	023552	000132*	/ ' Z
017141	000064	000012	000001	000000	000000	000000	000000*	4
100077	000100	000000	023534	023516	000132	017755	000077*	? 0 '\IN Z ?
000006	000001	023543	027275	000024	017202	020106	000006*	' . F
000001	023552	023507	000132	017714	000115	000025	000001*	' IG Z M
023561	000000	000132	000000	000143	000034	000001	023525*	' Z 'U
177777	000000	000000	000000	000000	000000	000000	177777*	

LOCATIONS 23600 THROUGH 23677

000000	000000	000000	000000	000000	000000	177777	000000*	
000000	000000	000000	000000	000000	177777	000000	000000*	
000000	000000	000000	000000	177777	000000	000000	000000*	
000000	000000	000000	177777	000000	000000	000000	000000*	
000000	000000	177777	000000	000000	000000	000000	000000*	

000000 000040 000001 000002 000003 000004 040005 040006\*      \*  
040007 040010 040011 040012 040041 000042 000043 000044\* \* \* \* \* ! " \* 3  
000045 000046 000047 000050 000051 000052 000053 140000\* x & ' ( ) \* +



# READER COMMENT SHEET

Manual Name: \_\_\_\_\_  
(Please Print)

Part Number: \_\_\_\_\_

We welcome your evaluation of this publication. Your comments and suggestions will help us improve our training materials. Please use additional pages if necessary.

Is this book technically accurate?

Did it meet your expectations?

Was it complete?

Is it easy to read and use?

Other comments?

\_\_\_\_\_  
FROM:

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_



**Training Coordinator/Technical Marketing**  
**Hewlett-Packard Co.**  
**11000 Wolfe Road**  
**Cupertino, California 95014**



